Docker

서비스?!

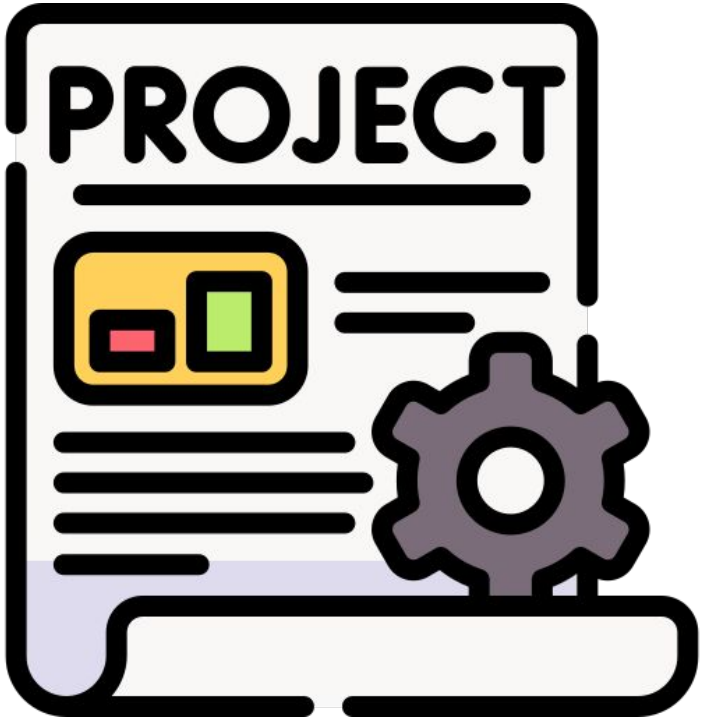
Google

NAVER

KAKAO TALK

BLOG

NAVER 카페


PROJECT

서비스를 운영하기 위해 필요한 것

기획

개발

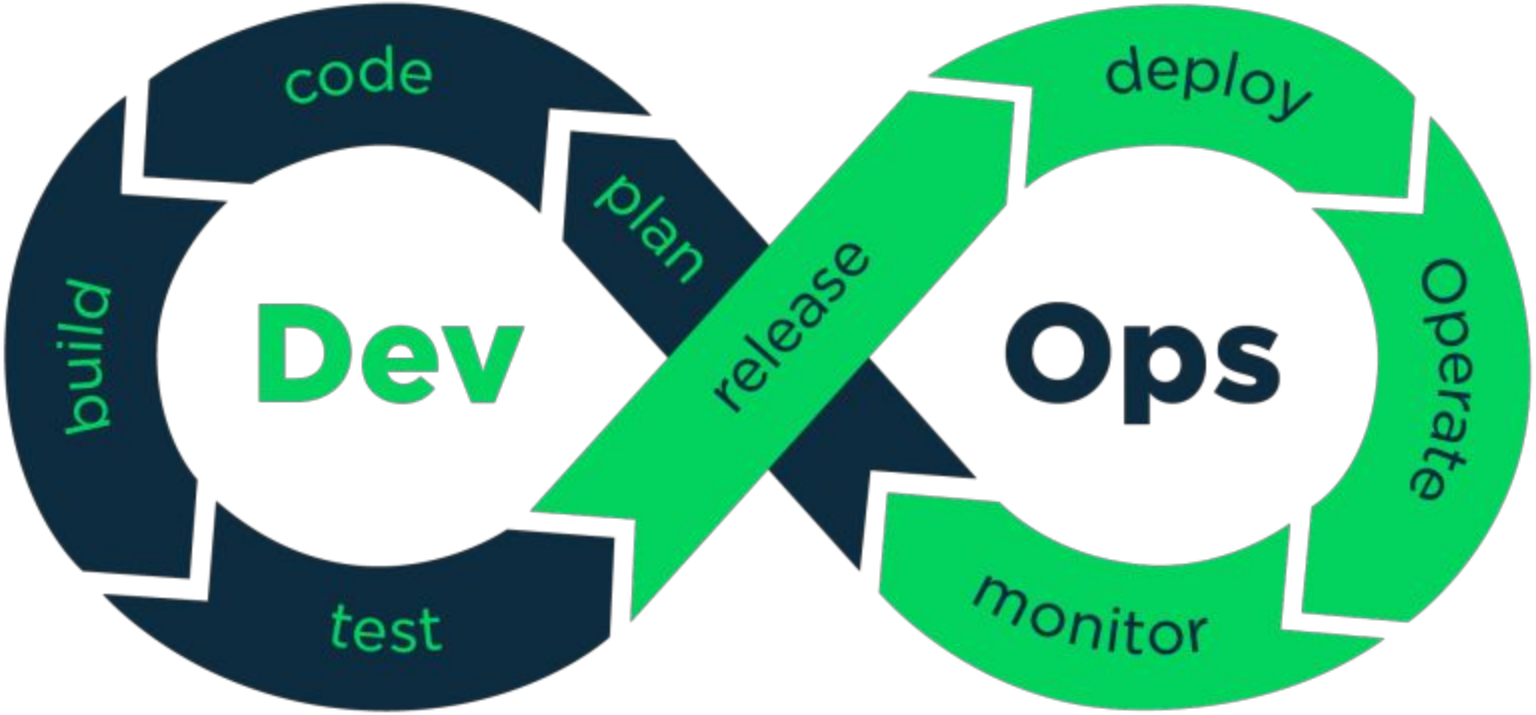
디자인

설계

서비스를 오랫동안
운영하기 위해 필요한 것

지속 가능성

(Sustainability)


code
plan
build
Dev
release
test
deploy
Ops
operate
monitor

반복적인 작업을

자동화!

자동화를 위한

추상화!!

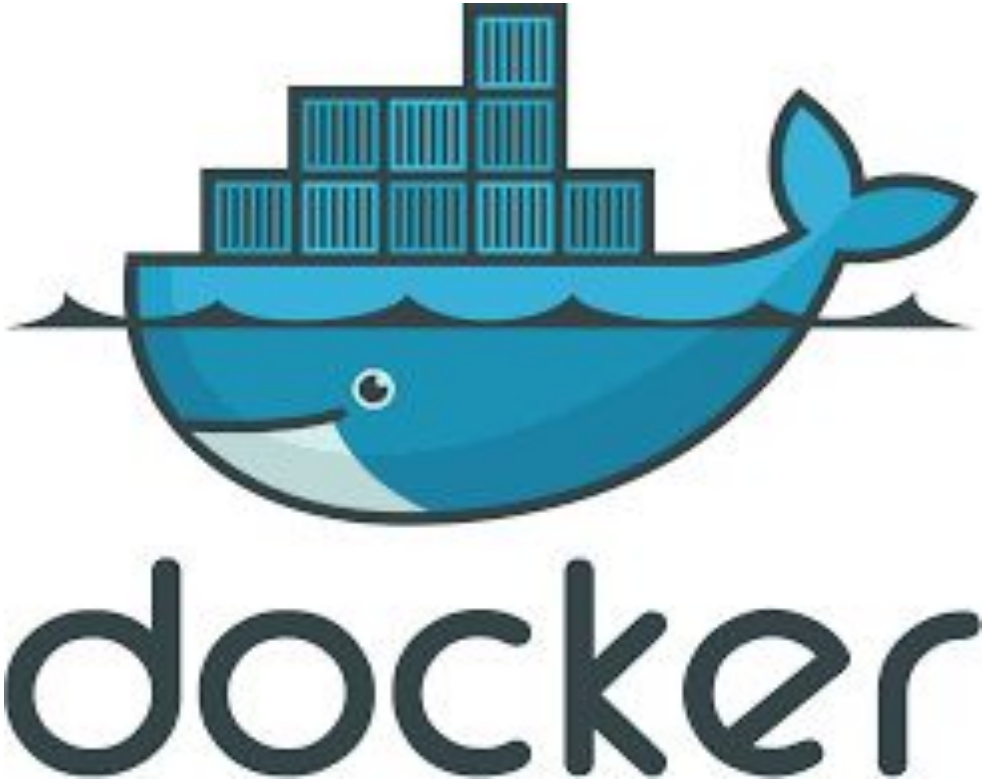
복잡한 자료, 모듈, 시스템 등으로부터
핵심적인 개념 또는 기능을
간추려 내는 것을 말한다.

무엇을?

하드웨어를!

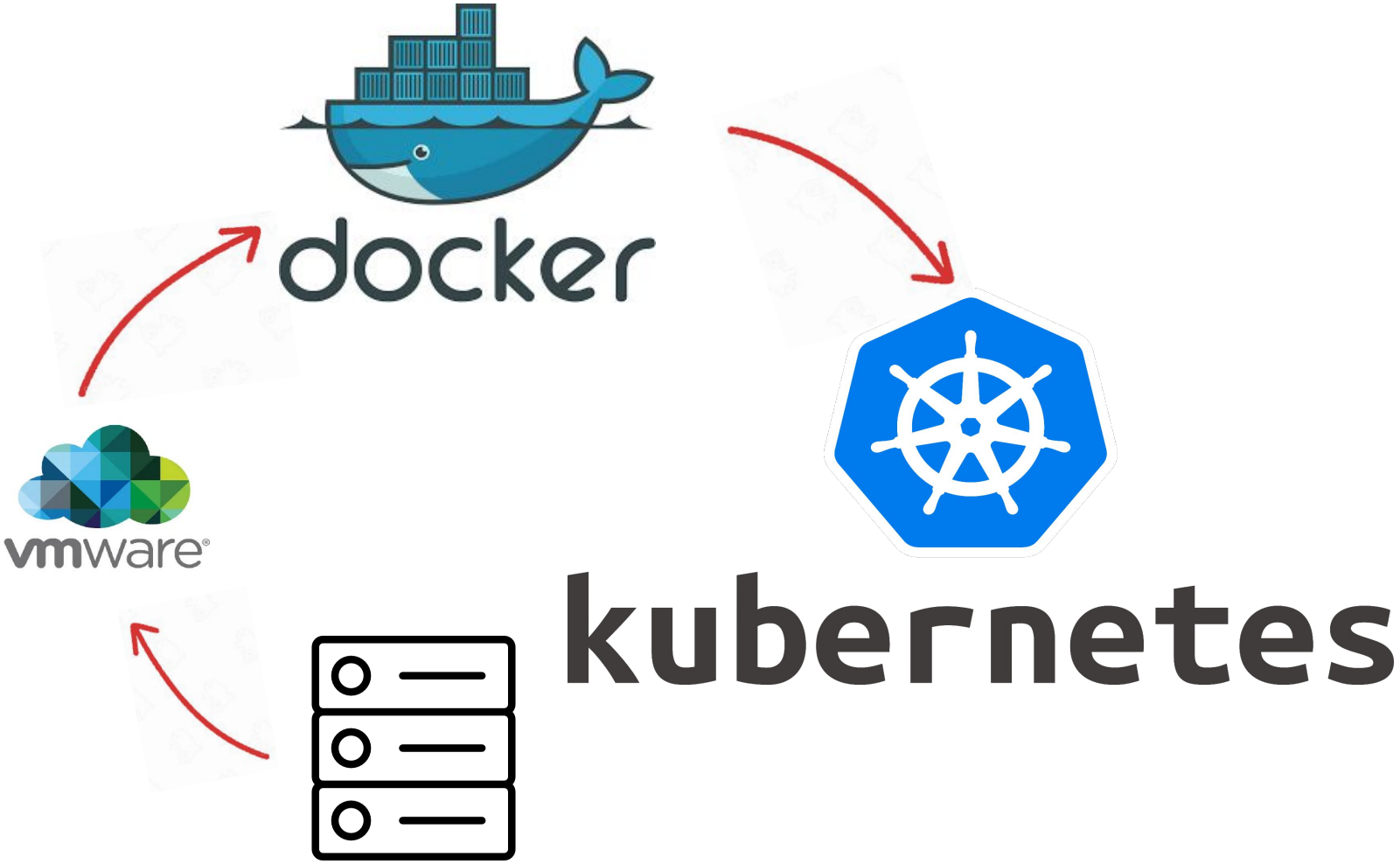

vmware®

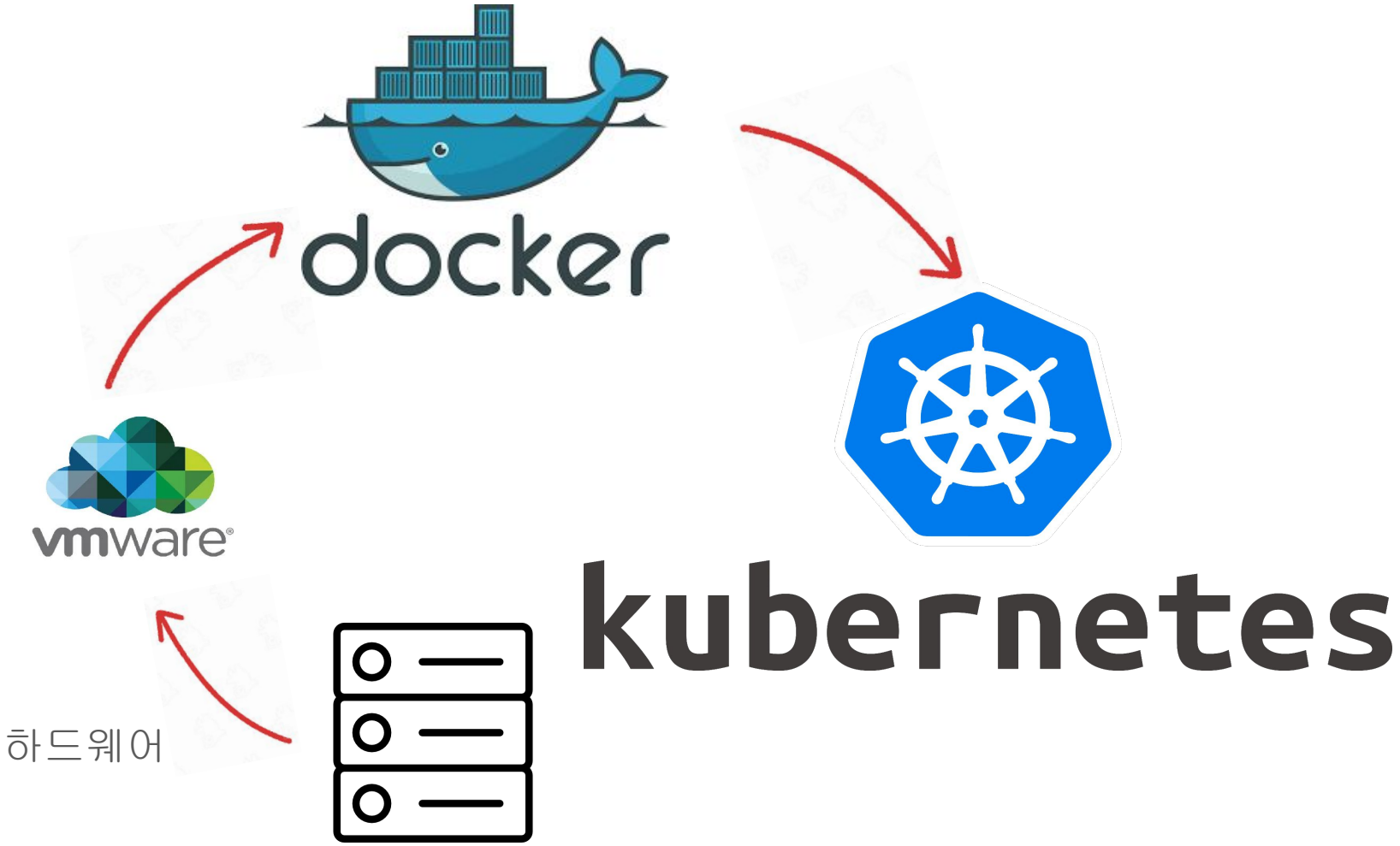
서비스를!

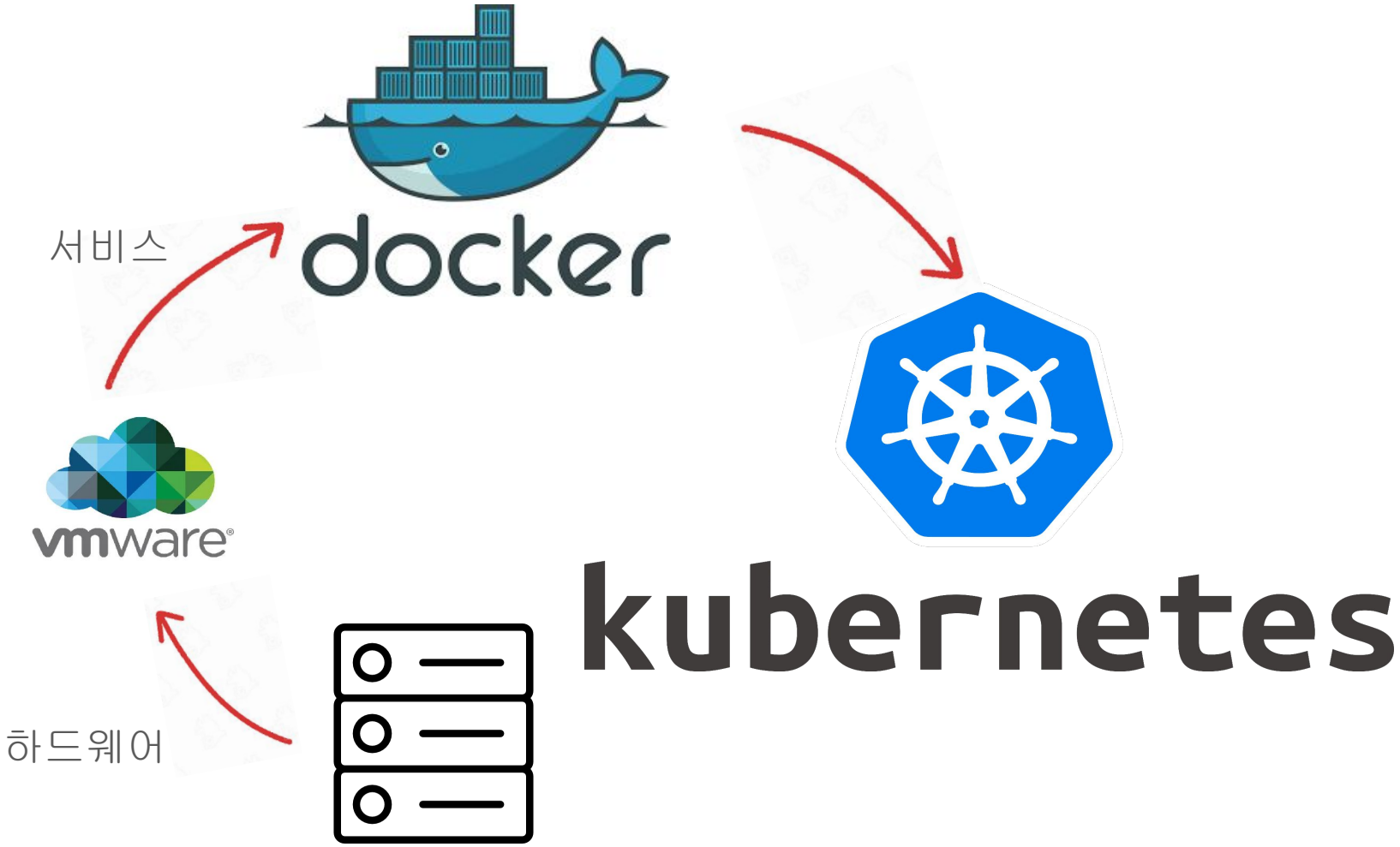

docker

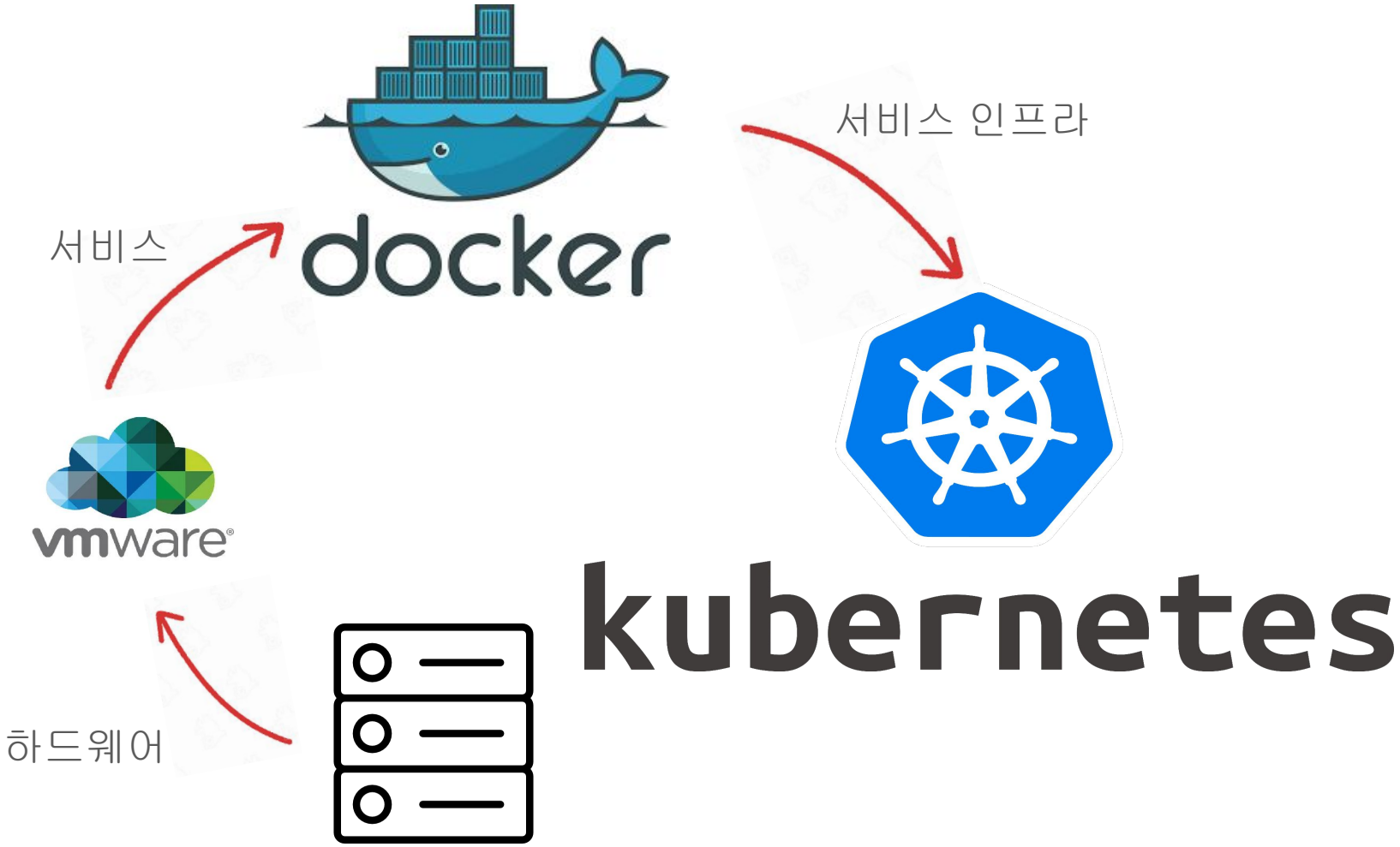
서비스 인프라를!

kubernetes

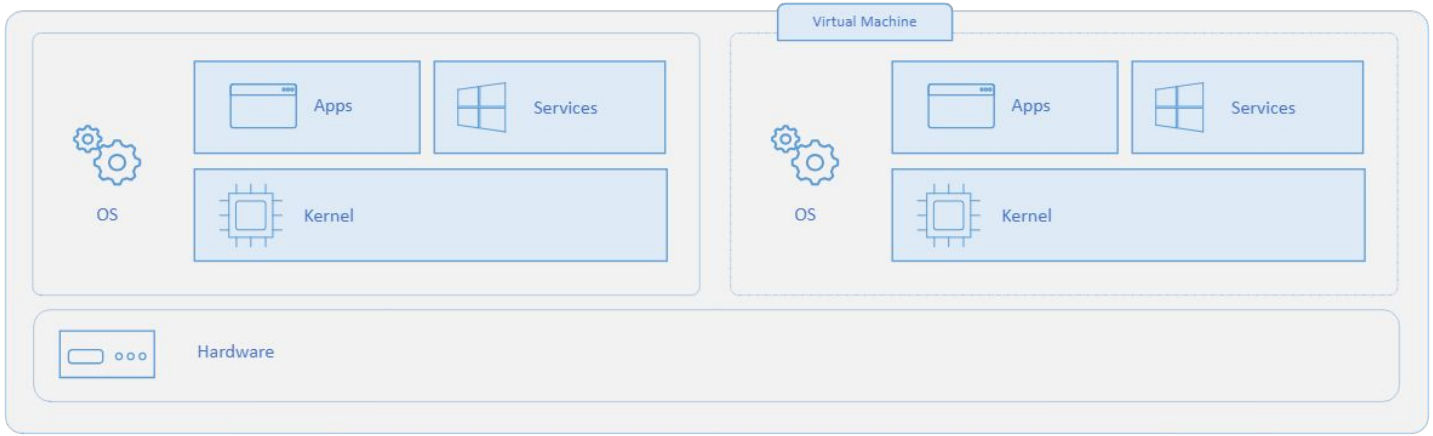

docker
kubernetes
vmware


docker
vmware
kubernetes
하드웨어

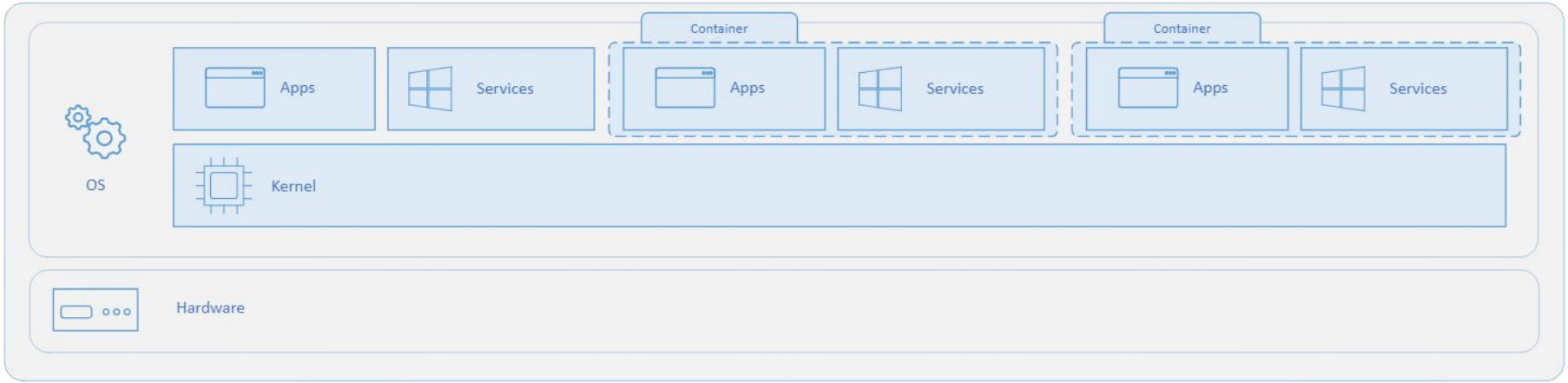

docker
서비스
vmware
kubernetes
하드웨어


docker
서비스 인프라
kubernetes
하드웨어
vmware
서비스

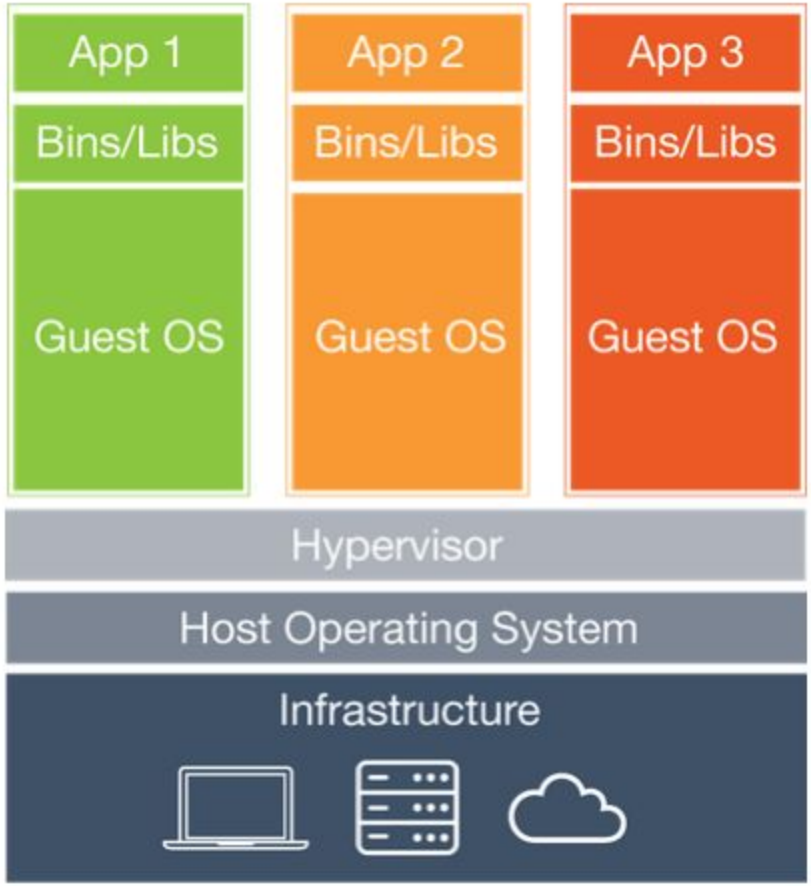
VM vs. Container

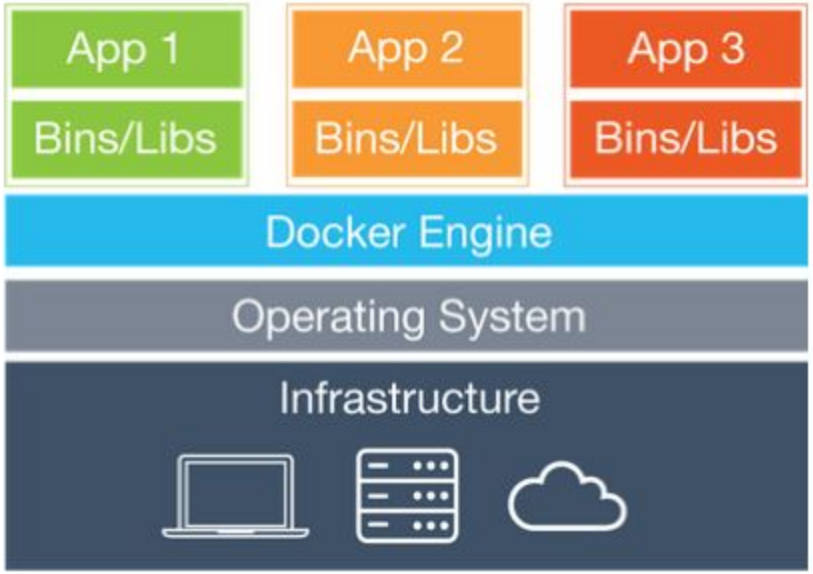


Virtual Machine

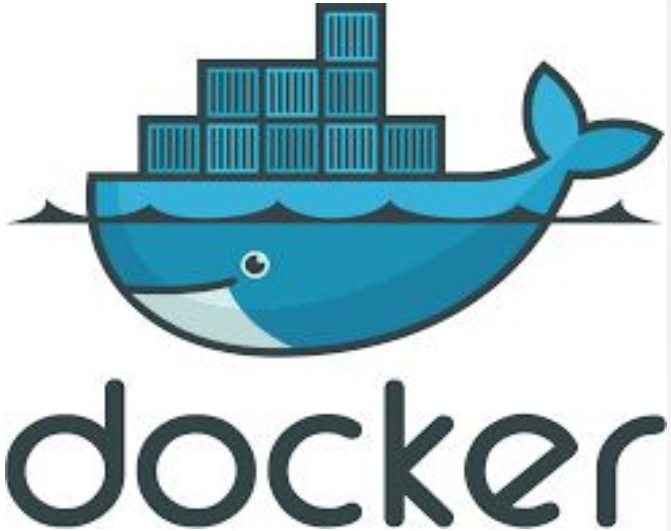

Container
Container
Apps
Services
Apps
Services
Apps
Services
OS
Kernel
Hardware

Container

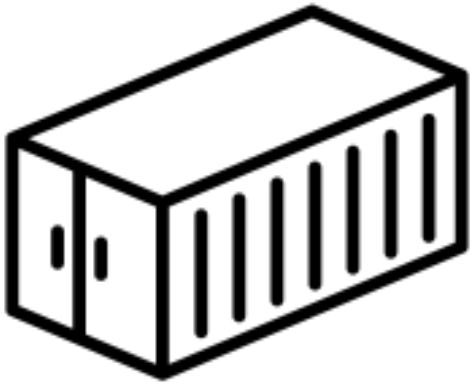

App 1
Bins/Libs
Guest OS
App 2
Bins/Libs
Guest OS
App 3
Bins/Libs
Guest OS
Hypervisor
Host Operating System
Infrastructure

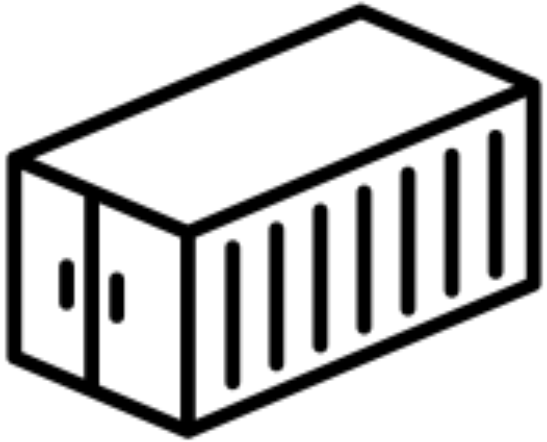

App 1
Bins/Libs
App 2
Bins/Libs
App 3
Bins/Libs
Docker Engine
Operating System
Infrastructure

시작하기에 앞서


docker

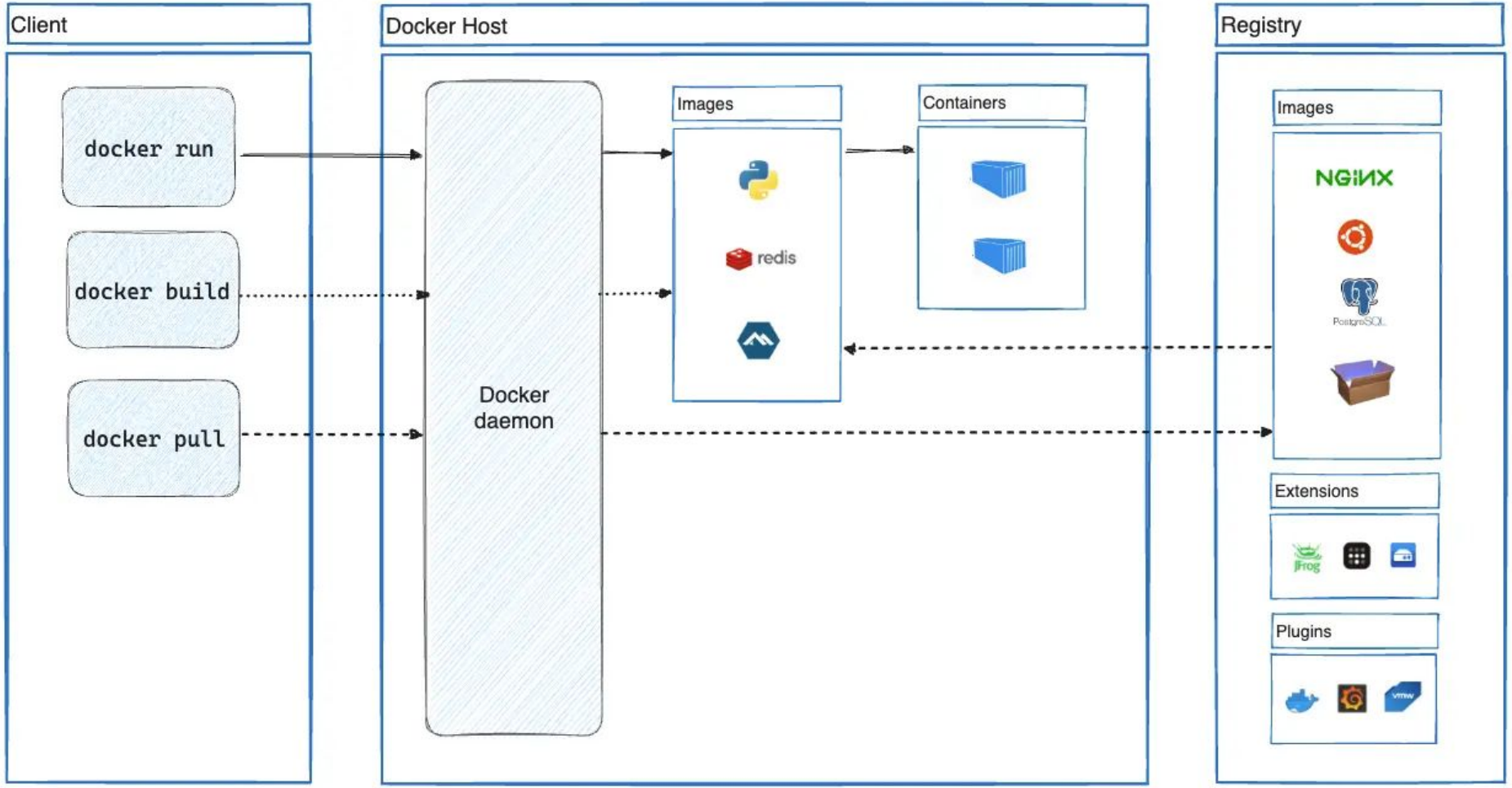
≠



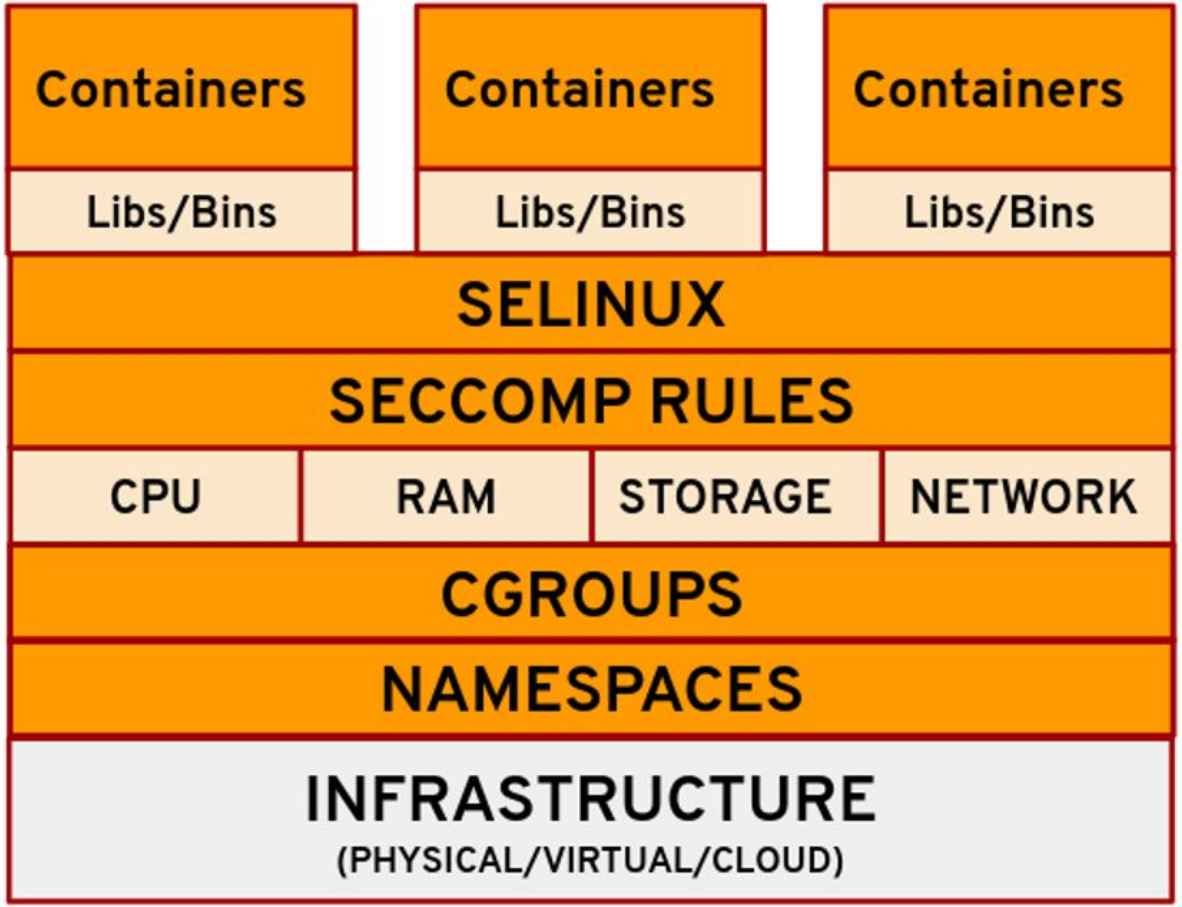


= 서비스의 추상화

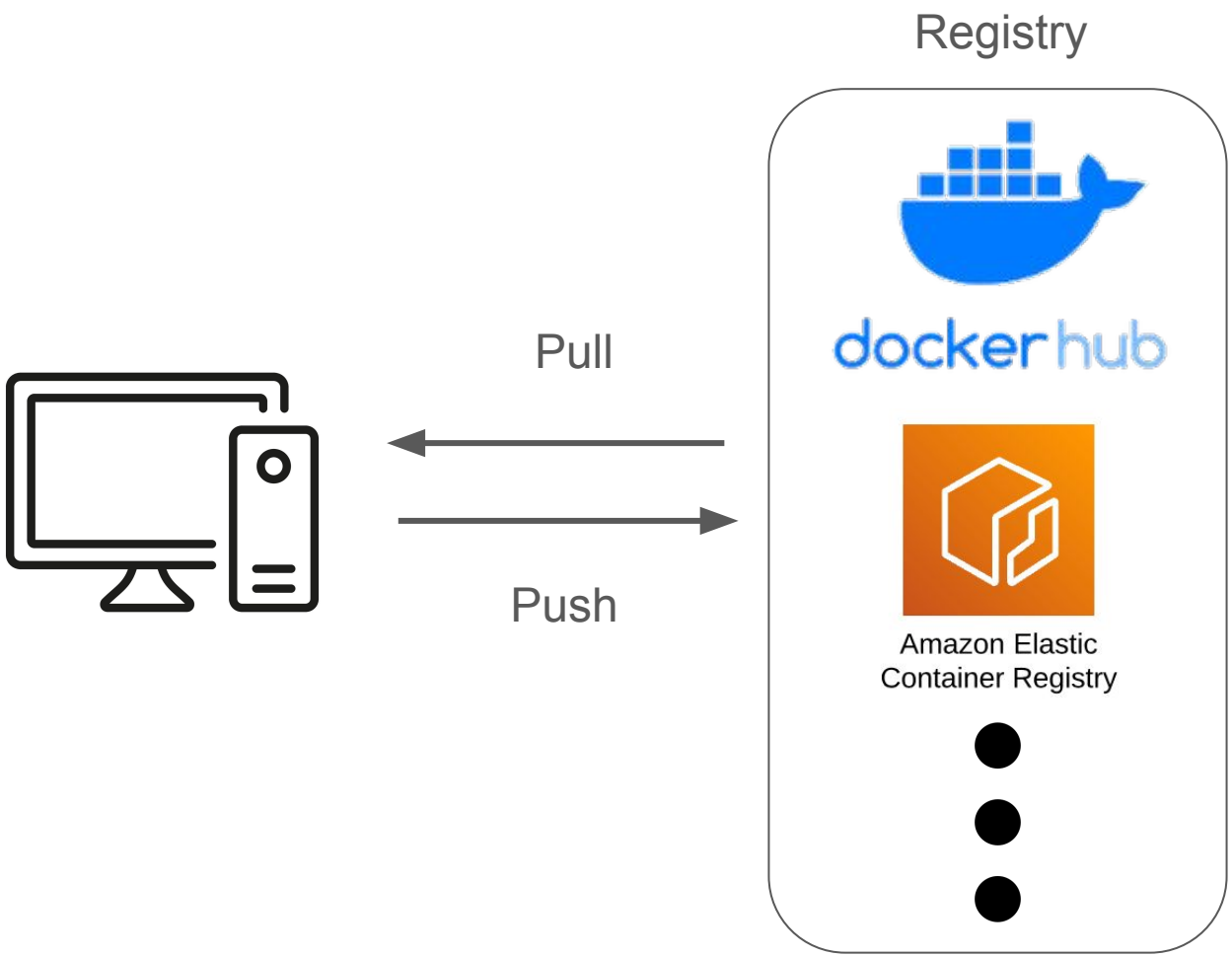
Docker 구조


Client
docker run
docker build
docker pull
Docker Host
Docker daemon
Images
redis
Containers
Registry
Images
NGINX
PostgreSQL
Extensions
Plugins

컨테이너 구조

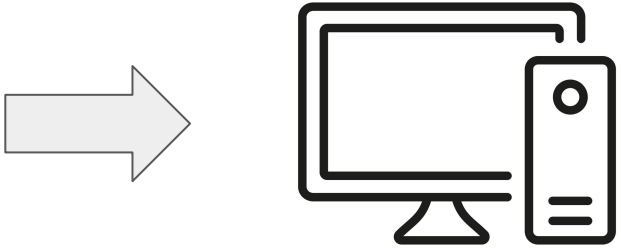

Containers
Containers
Containers
Libs/Bins
Libs/Bins
Libs/Bins
SELINUX
SECCOMP RULES
CPU
RAM
STORAGE
NETWORK
CGROUPS
NAMESPACES
INFRASTRUCTURE
(PHYSICAL/VIRTUAL/CLOUD)

Docker CLI
- 이미지 관리

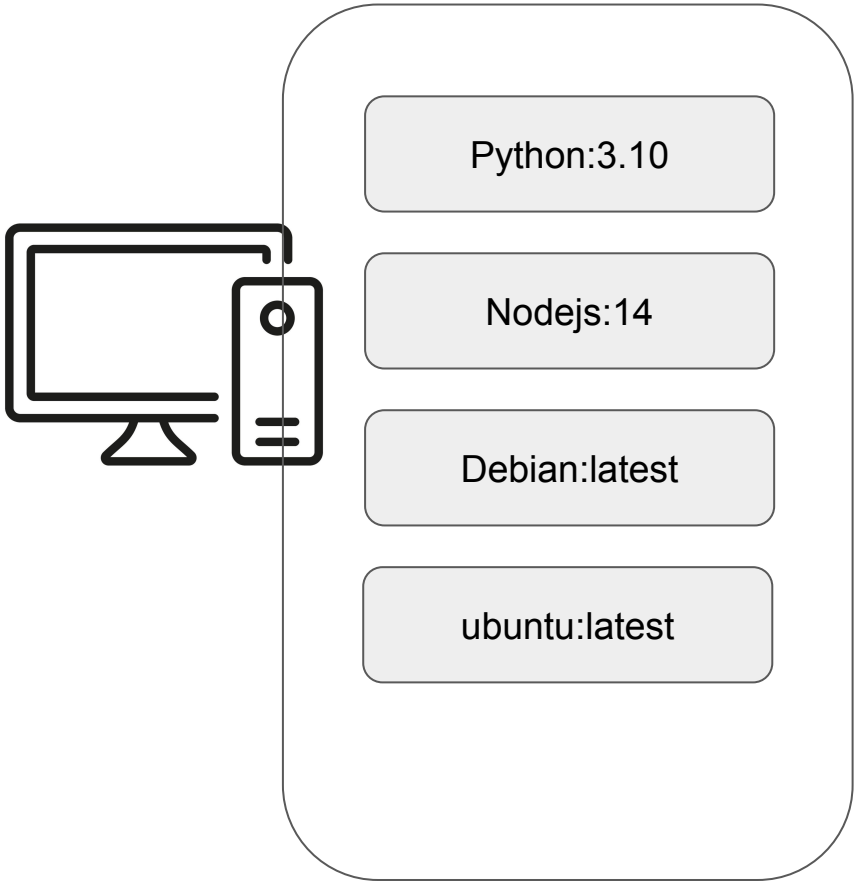

Registry
Pull
Push
docker hub
Amazon Elastic
Container Registry

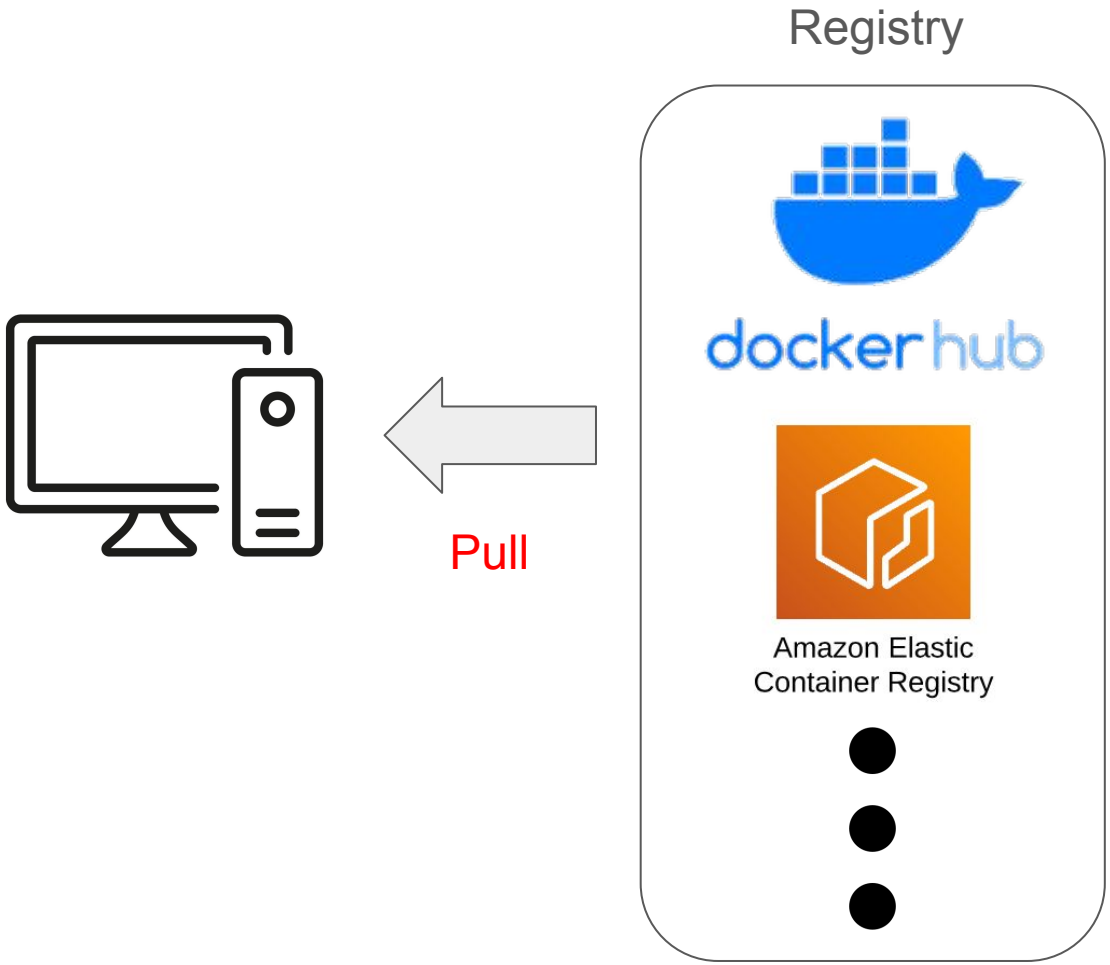
Docker CLI
- 컨테이너 관리

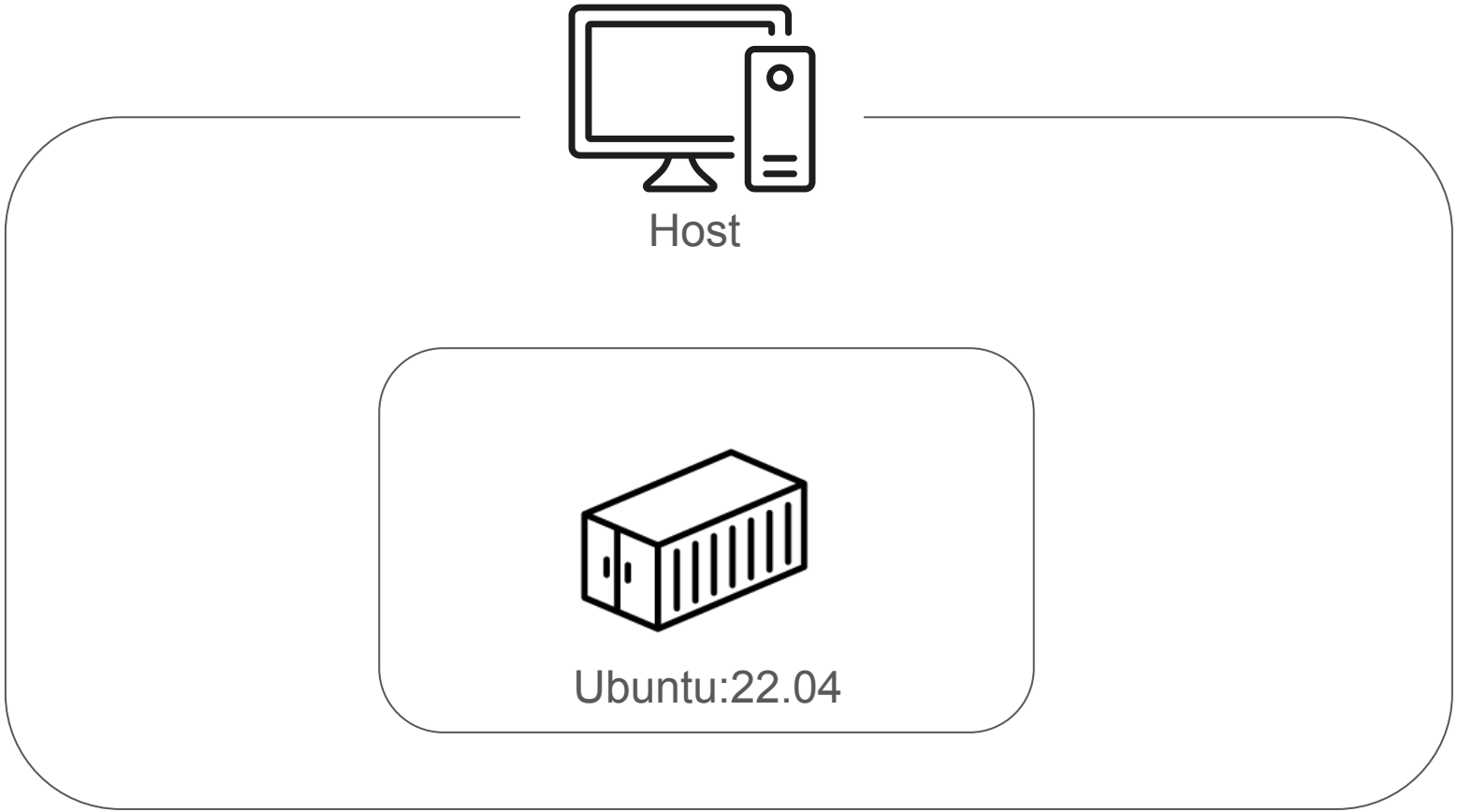
docker run
ubuntu:22.04



docker run
ubuntu:22.04

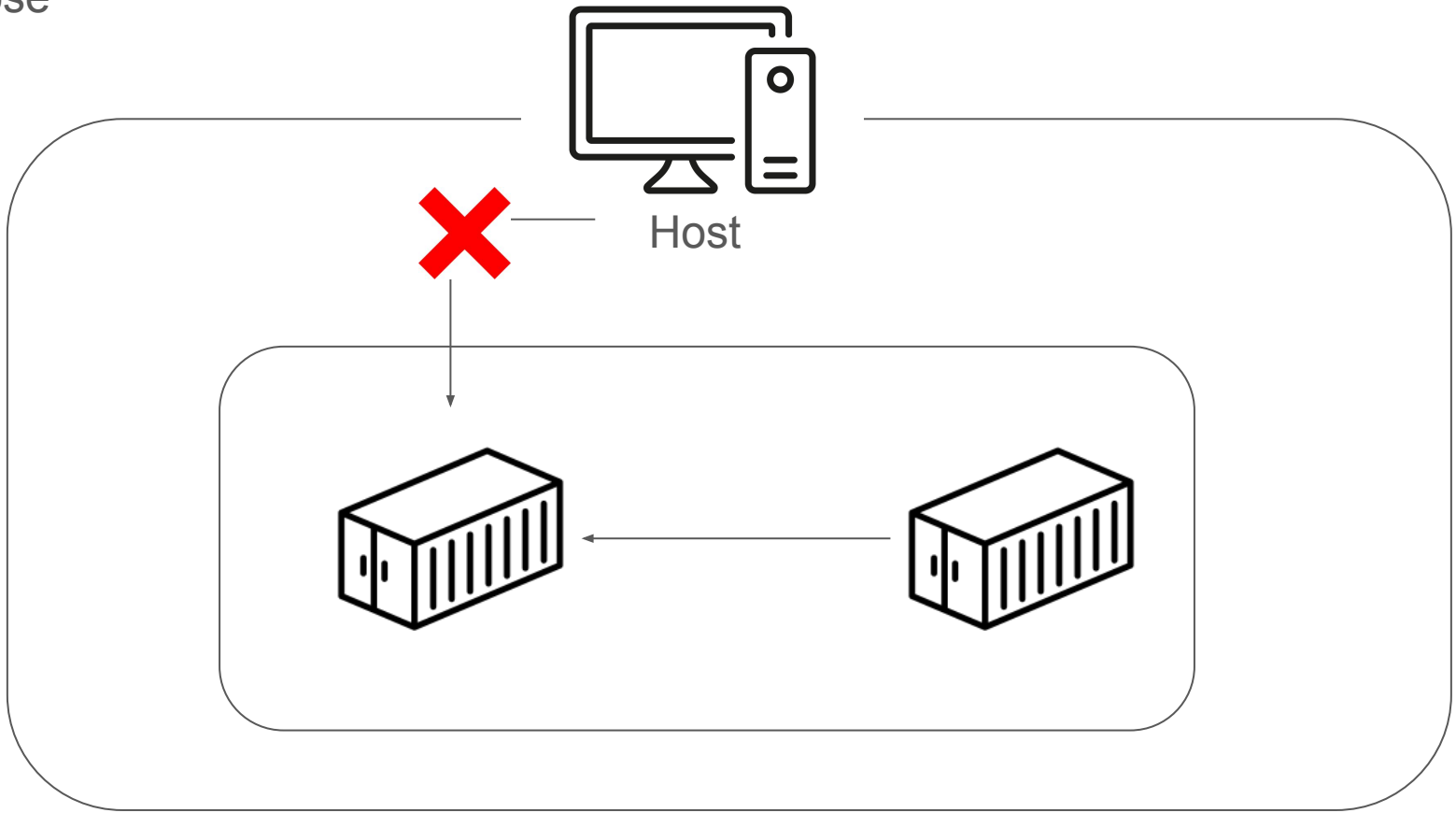



Registry
dockerhub
Amazon Elastic
Container Registry
Pull

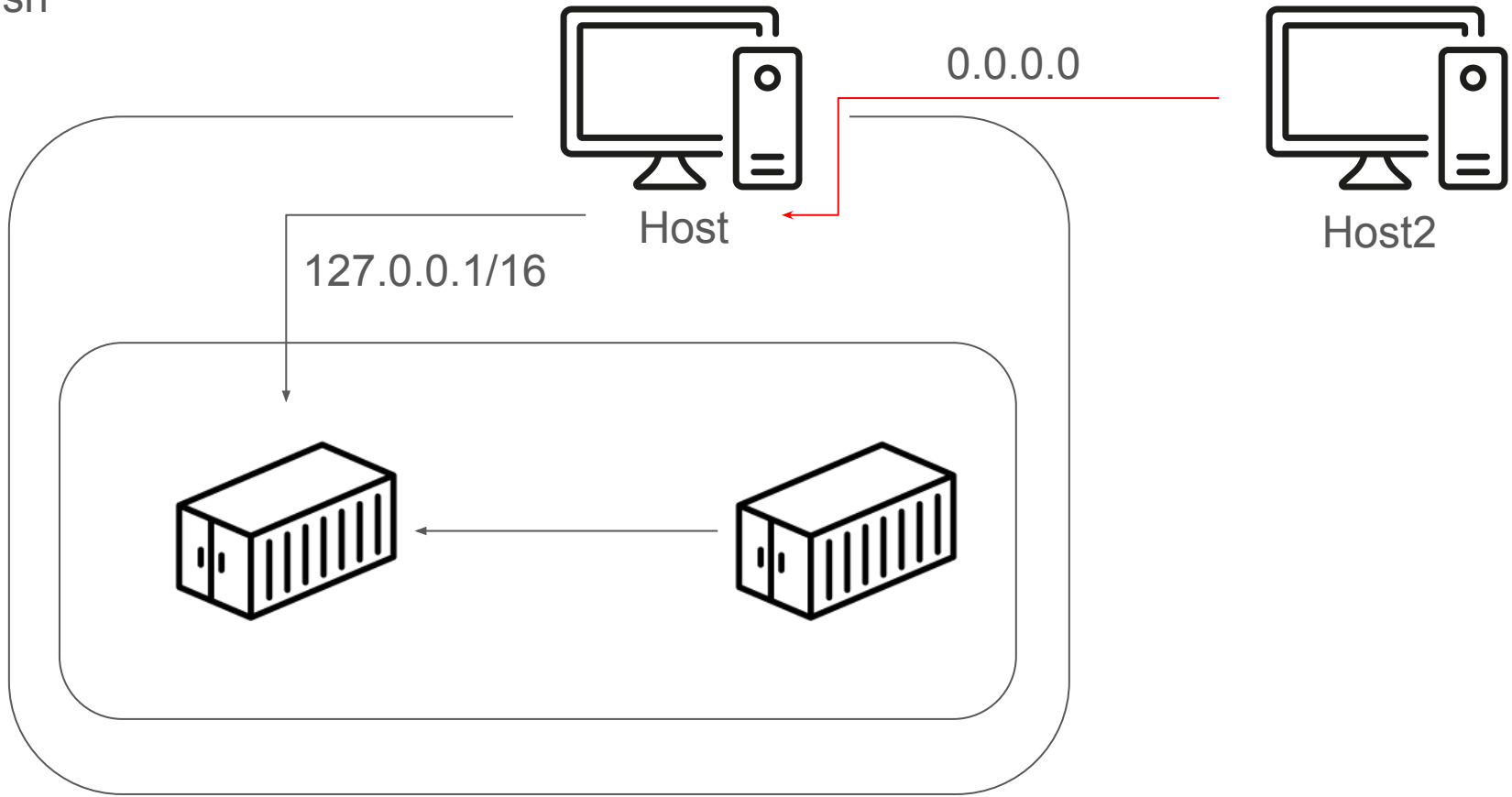

Host
Ubuntu:22.04

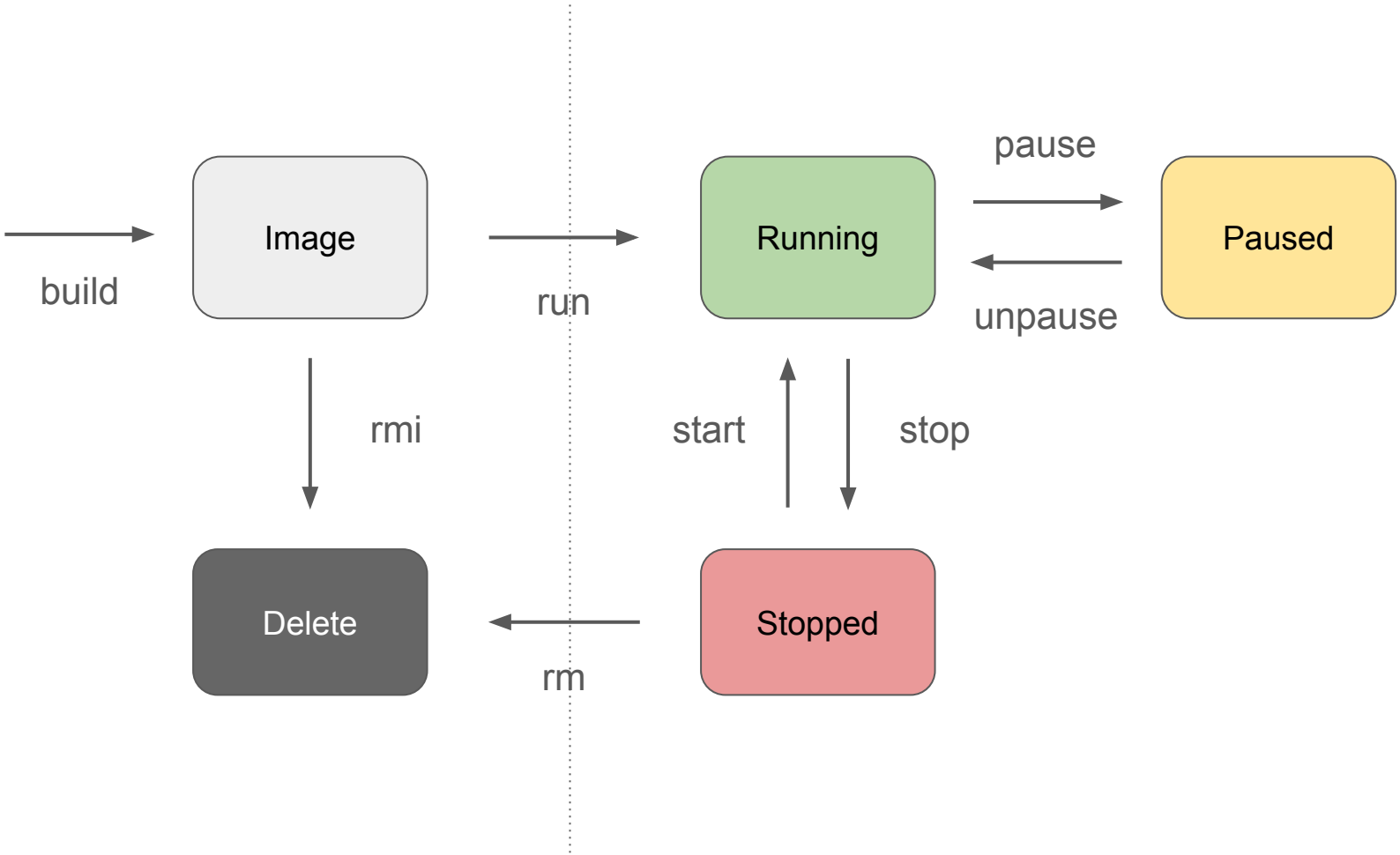
Expose Vs. Publish

Expose



Publish




build
Image
run
Running
pause
Paused
unpause
rmi
start
stop
Delete
rm
Stopped

Docker CLI
- 이미지 생성

Dockerfile

```
ARG TAG=22.04

FROM ubuntu:$TAG

LABEL PROJECT=cloudwave

ENV STAGE=prod

ADD https://docs.docker.com/engine/reference/builder /sample.html

COPY ./jar /src/jar

USER root

WORKDIR /root

RUN apt-get update && apt-get upgrade

CMD ["/bin/bash"]

ENTRYPOINT [""]
```

```
ARG TAG=22.04

FROM ubuntu:$TAG

LABEL PROJECT=cloudwave

ENV STAGE=prod

ADD https://docs.docker.com/engine/reference/builder /sample.html

COPY ./jar /src/jar

USER root

WORKDIR /root

RUN apt-get update && apt-get upgrade

CMD ["/bin/bash"]

ENTRYPOINT [""]
```

```
ARG TAG=22.04

FROM ubuntu:$TAG

LABEL PROJECT=cloudwave

ENV STAGE=prod

ADD https://docs.docker.com/engine/reference/builder /sample.html

COPY ./jar /src/jar

USER root

WORKDIR /root

RUN apt-get update && apt-get upgrade

CMD ["/bin/bash"]

ENTRYPOINT [""]
```

```
ARG TAG=22.04

FROM ubuntu:$TAG

LABEL PROJECT=cloudwave

ENV STAGE=prod

ADD https://docs.docker.com/engine/reference/builder /sample.html

COPY ./jar /src/jar

USER root

WORKDIR /root

RUN apt-get update && apt-get upgrade

CMD ["/bin/bash"]

ENTRYPOINT [""]
```

```
ARG TAG=22.04

FROM ubuntu:$TAG

LABEL PROJECT=cloudwave

ENV STAGE=prod

ADD https://docs.docker.com/engine/reference/builder /sample.html

COPY ./jar /src/jar

USER root

WORKDIR /root

RUN apt-get update && apt-get upgrade

CMD ["/bin/bash"]

ENTRYPOINT [""]
```

```
ARG TAG=22.04

FROM ubuntu:$TAG

LABEL PROJECT=cloudwave

ENV STAGE=prod

ADD https://docs.docker.com/engine/reference/builder /sample.html

COPY ./jar /src/jar

USER root

WORKDIR /root

RUN apt-get update && apt-get upgrade

CMD ["/bin/bash"]

ENTRYPOINT [""]
```

```
ARG TAG=22.04

FROM ubuntu:$TAG

LABEL PROJECT=cloudwave

ENV STAGE=prod

ADD https://docs.docker.com/engine/reference/builder /sample.html

COPY ./jar /src/jar

USER root

WORKDIR /root

RUN apt-get update && apt-get upgrade

CMD ["/bin/bash"]

ENTRYPOINT [""]
```

Add Vs. Copy

```
ARG TAG=22.04

FROM ubuntu:$TAG

LABEL PROJECT=cloudwave

ENV STAGE=prod

ADD https://docs.docker.com/engine/reference/builder /sample.html

COPY ./jar /src/jar

USER root

WORKDIR /root

RUN apt-get update && apt-get upgrade

CMD ["/bin/bash"]

ENTRYPOINT [""]
```

```
ARG TAG=22.04

FROM ubuntu:$TAG

LABEL PROJECT=cloudwave

ENV STAGE=prod

ADD https://docs.docker.com/engine/reference/builder /sample.html

COPY ./jar /src/jar

USER root

WORKDIR /root

RUN apt-get update && apt-get upgrade

CMD ["/bin/bash"]

ENTRYPOINT [""]
```

```
ARG TAG=22.04

FROM ubuntu:$TAG

LABEL PROJECT=cloudwave

ENV STAGE=prod

ADD https://docs.docker.com/engine/reference/builder /sample.html

COPY ./jar /src/jar

USER root

WORKDIR /root

RUN apt-get update && apt-get upgrade

CMD ["/bin/bash"]

ENTRYPOINT [""]
```

```
ARG TAG=22.04

FROM ubuntu:$TAG

LABEL PROJECT=cloudwave

ENV STAGE=prod

ADD https://docs.docker.com/engine/reference/builder /sample.html

COPY ./jar /src/jar

USER root

WORKDIR /root

RUN apt-get update && apt-get upgrade

CMD ["/bin/bash"]

ENTRYPOINT [""]
```

RUN Vs. CMD Vs. ENTRYPOINT

RUN Vs. CMD Vs. ENTRYPOINT

이미지를 빌드하는 과정에서 실행

ex) 필요한 Package 설치

RUN Vs. CMD Vs. ENTRYPOINT

컨테이너를 실행하는 순간 실행

ex) 서비스 실행을 위한 스크립트,

…

결정적인 차이점!

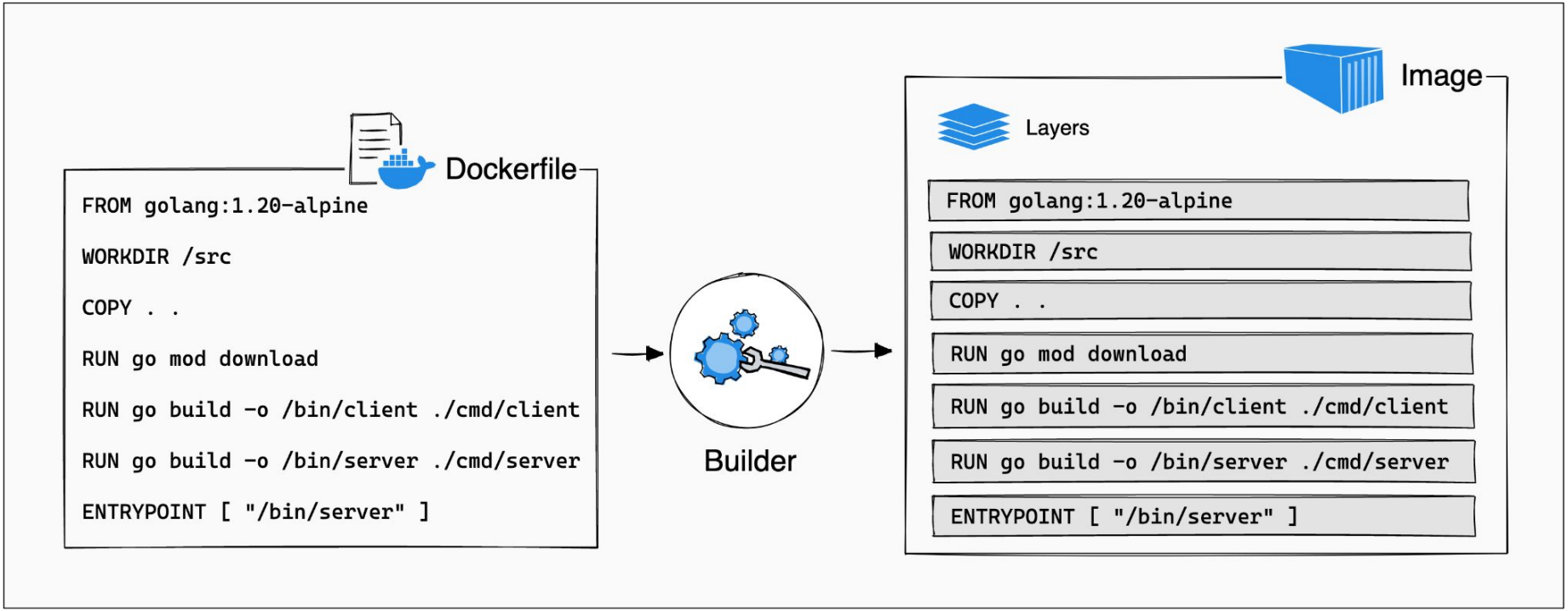
CMD는 `docker run`시에

변경이 가능!

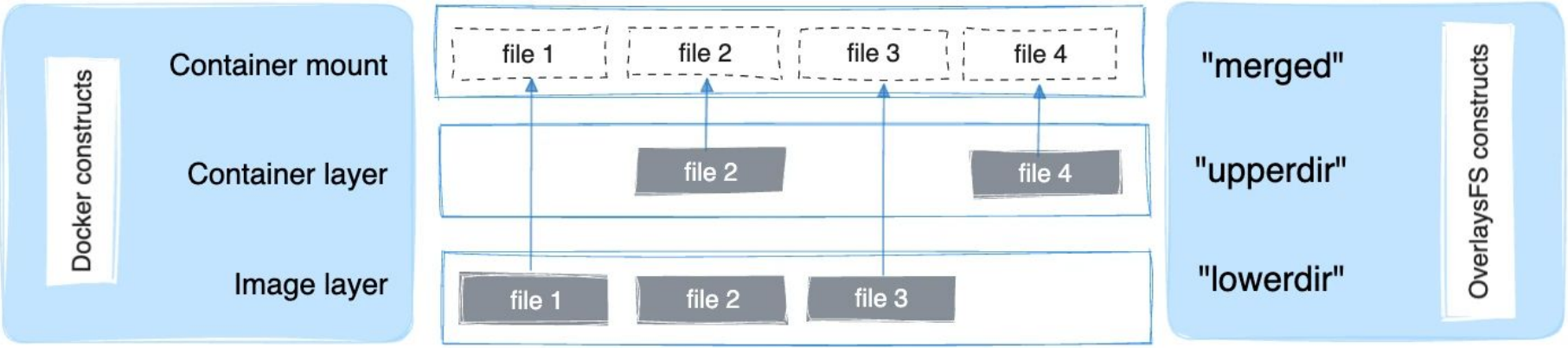
ENTRYPOINT는

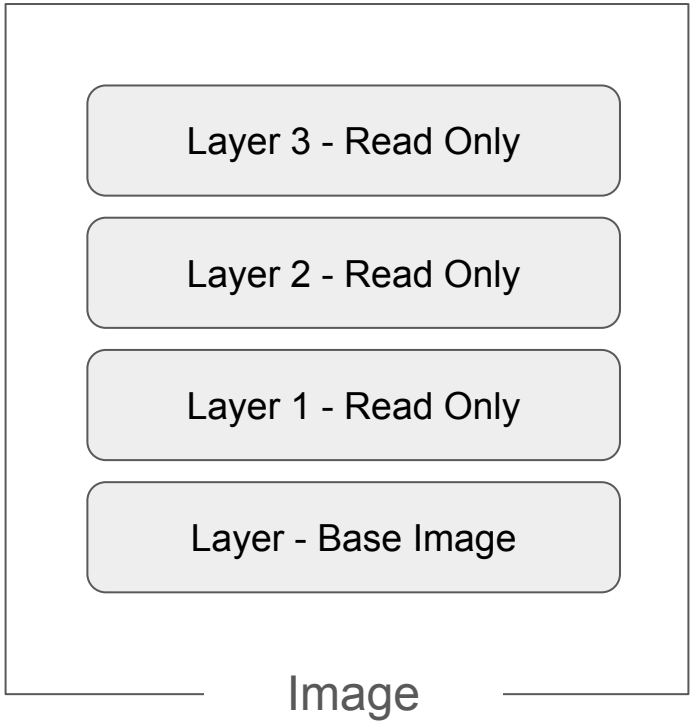
변경이 불가능!

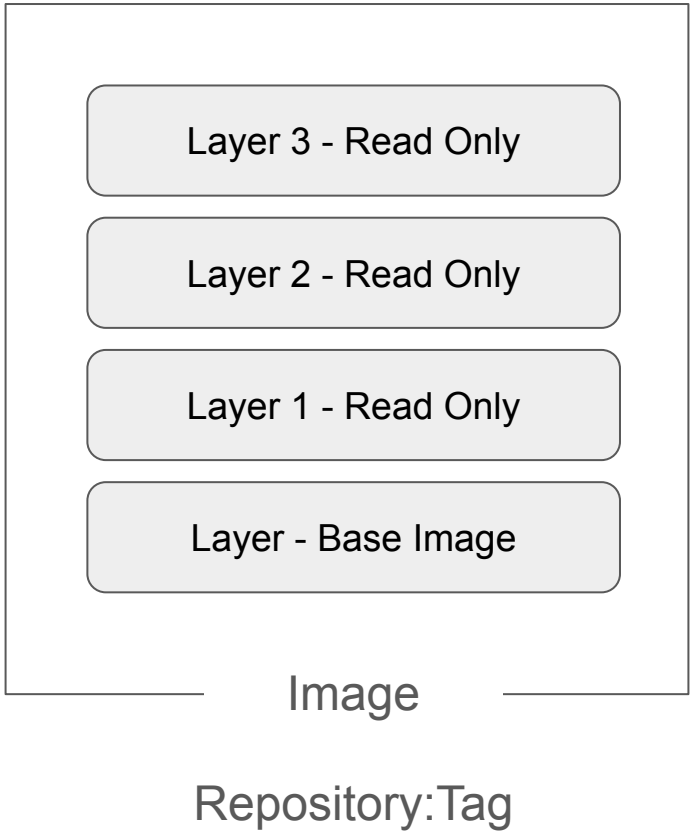
Image & Layer

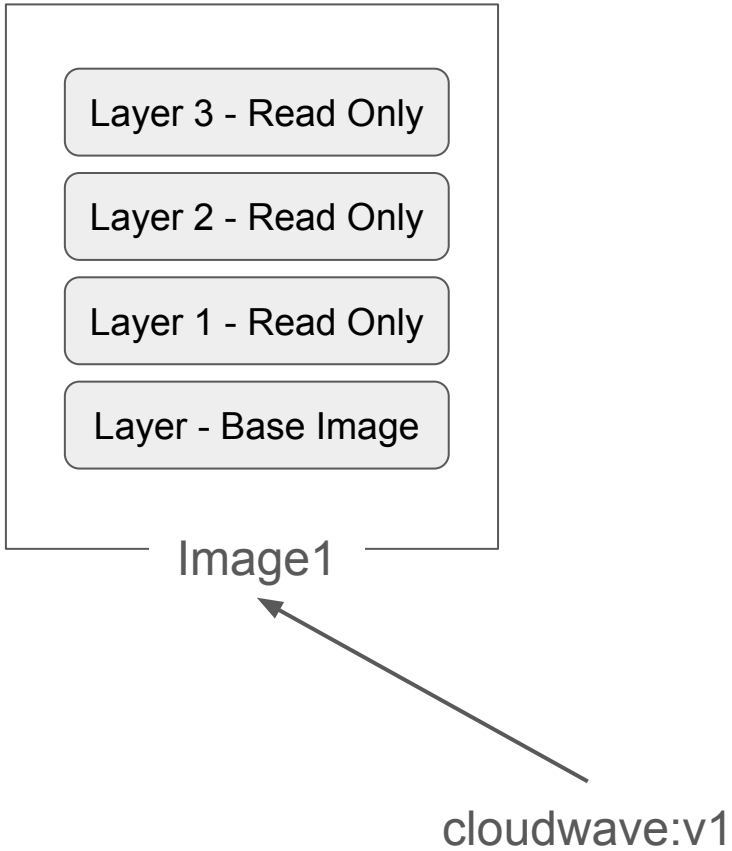

Dockerfile
FROM golang:1.20-alpine
WORKDIR /src
COPY . .
RUN go mod download
RUN go build -o /bin/client ./cmd/client
RUN go build -o /bin/server ./cmd/server
ENTRYPOINT ["/bin/server"]
Builder
Image
Layers
FROM golang:1.20-alpine
WORKDIR /src
COPY . .
RUN go mod download
RUN go build -o /bin/client ./cmd/client
RUN go build -o /bin/server ./cmd/server
ENTRYPOINT ["/bin/server"]

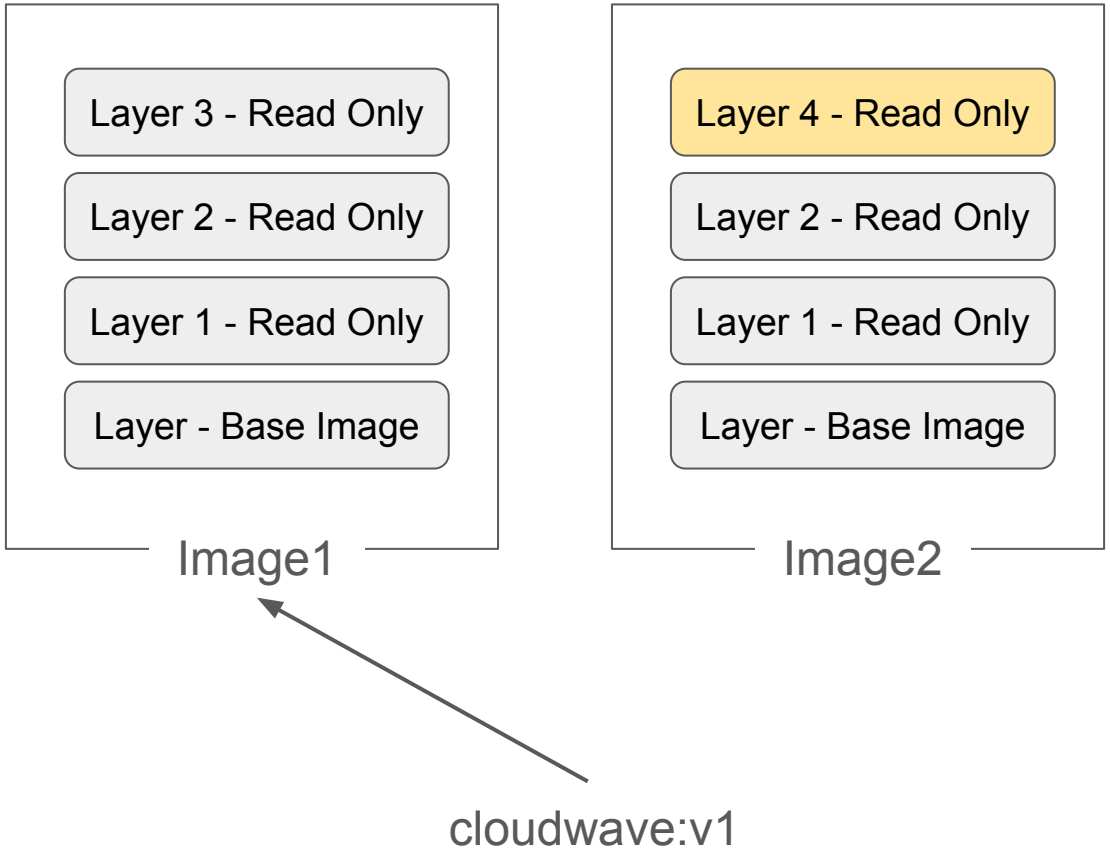
OverlayFS

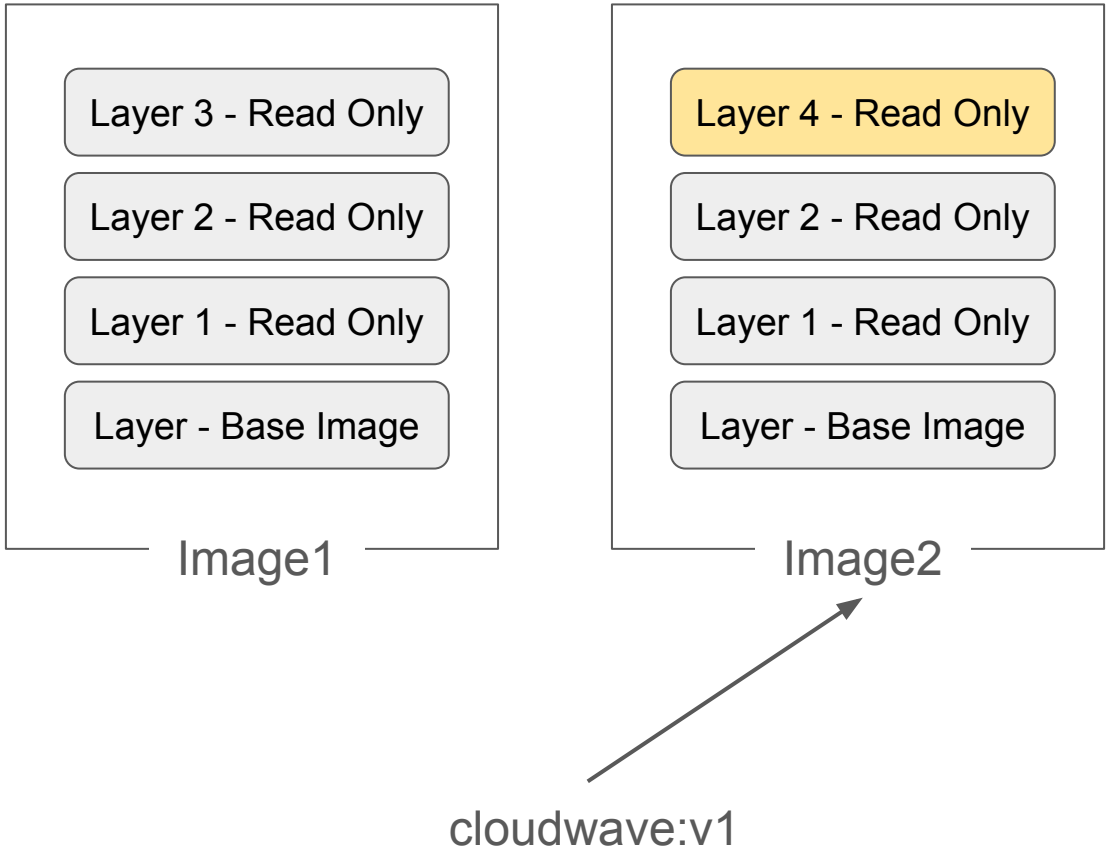

Docker constructs
Container mount
Container layer
Image layer
file 1
file 2
file 3
file 4
file 2
file 4
file 1
file 2
file 3
"merged"
"upperdir"
"lowerdir"
OverlaysFS constructs

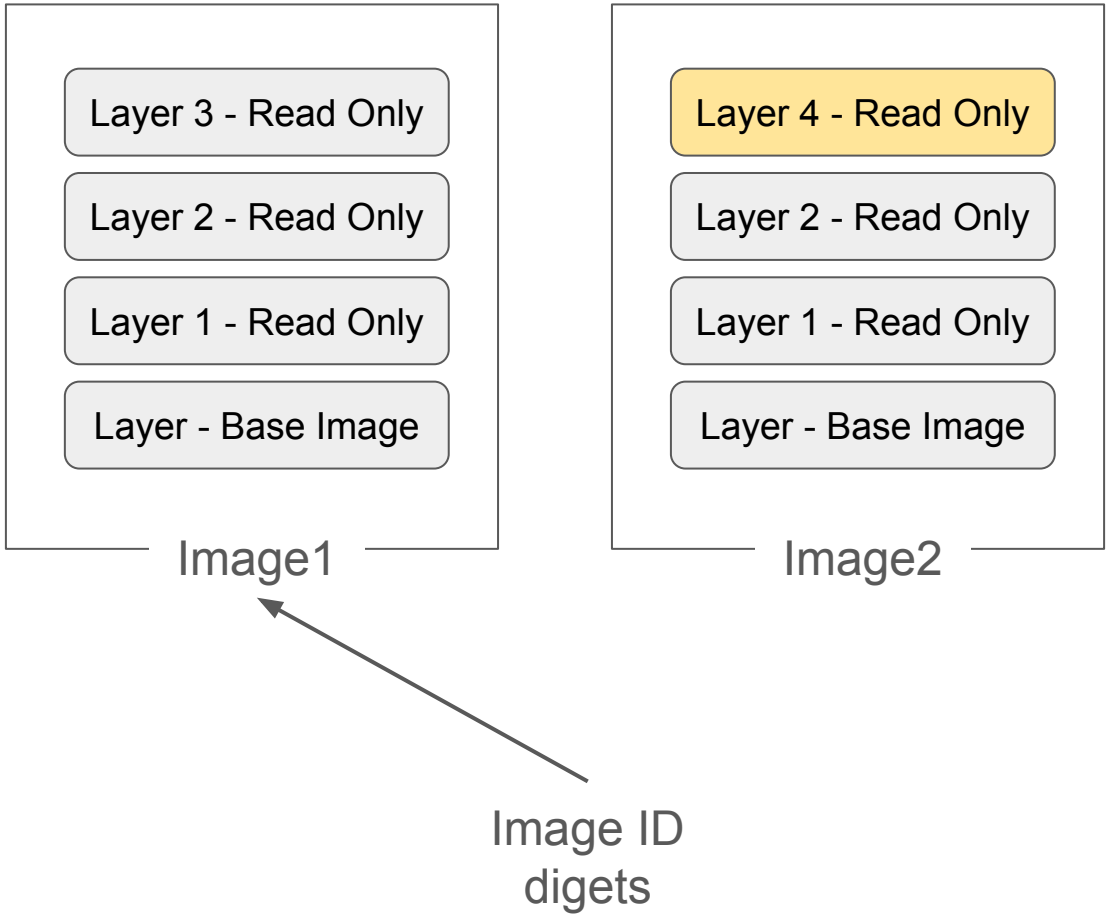

Layer 3 - Read Only
Layer 2 - Read Only
Layer 1 - Read Only
Layer - Base Image
Image


Layer 3 - Read Only
Layer 2 - Read Only
Layer 1 - Read Only
Layer - Base Image
Image
Repository:Tag

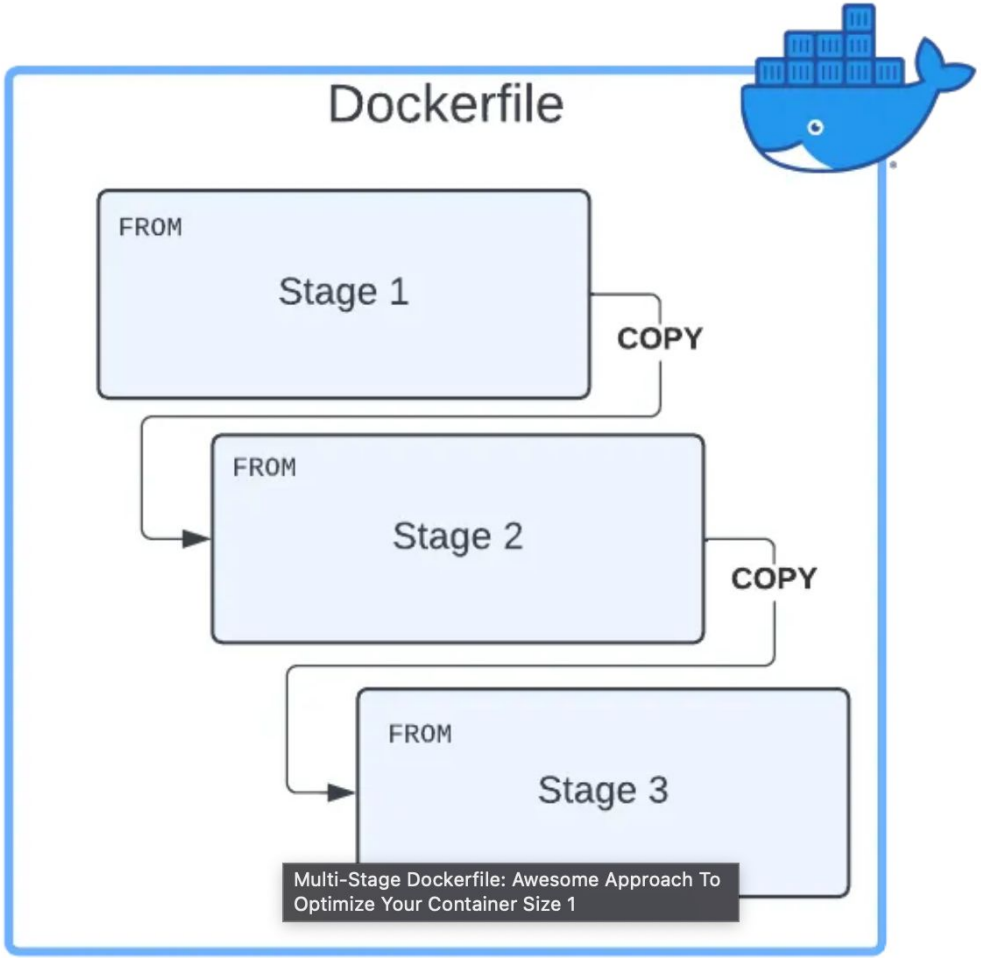

Layer 3 - Read Only
Layer 2 - Read Only
Layer 1 - Read Only
Layer - Base Image
Image1
cloudwave:v1


Layer 3 - Read Only
Layer 2 - Read Only
Layer 1 - Read Only
Layer - Base Image
Image1
Layer 4 - Read Only
Layer 2 - Read Only
Layer 1 - Read Only
Layer - Base Image
Image2
cloudwave:v1


Layer 3 - Read Only
Layer 2 - Read Only
Layer 1 - Read Only
Layer - Base Image
Image1
Layer 4 - Read Only
Layer 2 - Read Only
Layer 1 - Read Only
Layer - Base Image
Image2
cloudwave:v1


Layer 3 - Read Only
Layer 2 - Read Only
Layer 1 - Read Only
Layer - Base Image
Image1
Layer 4 - Read Only
Layer 2 - Read Only
Layer 1 - Read Only
Layer - Base Image
Image2
Image ID
digets

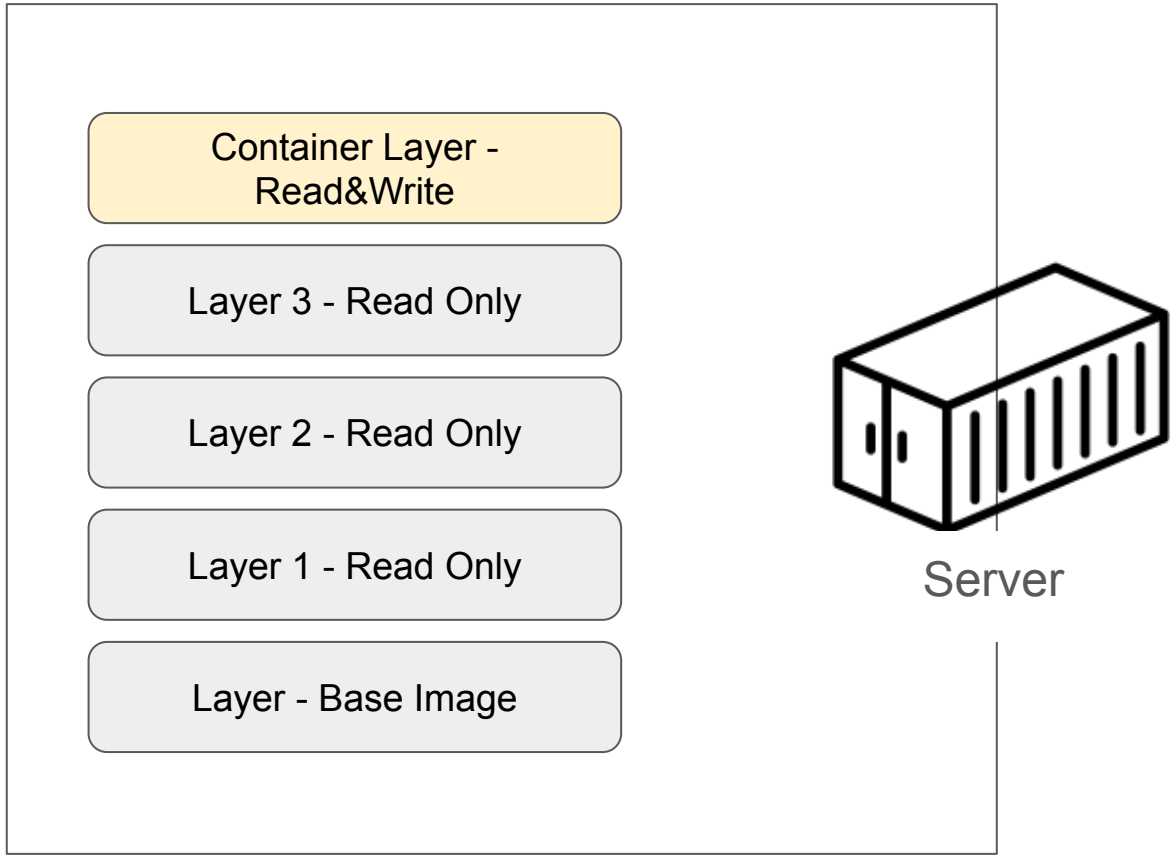
Multi-Stage build

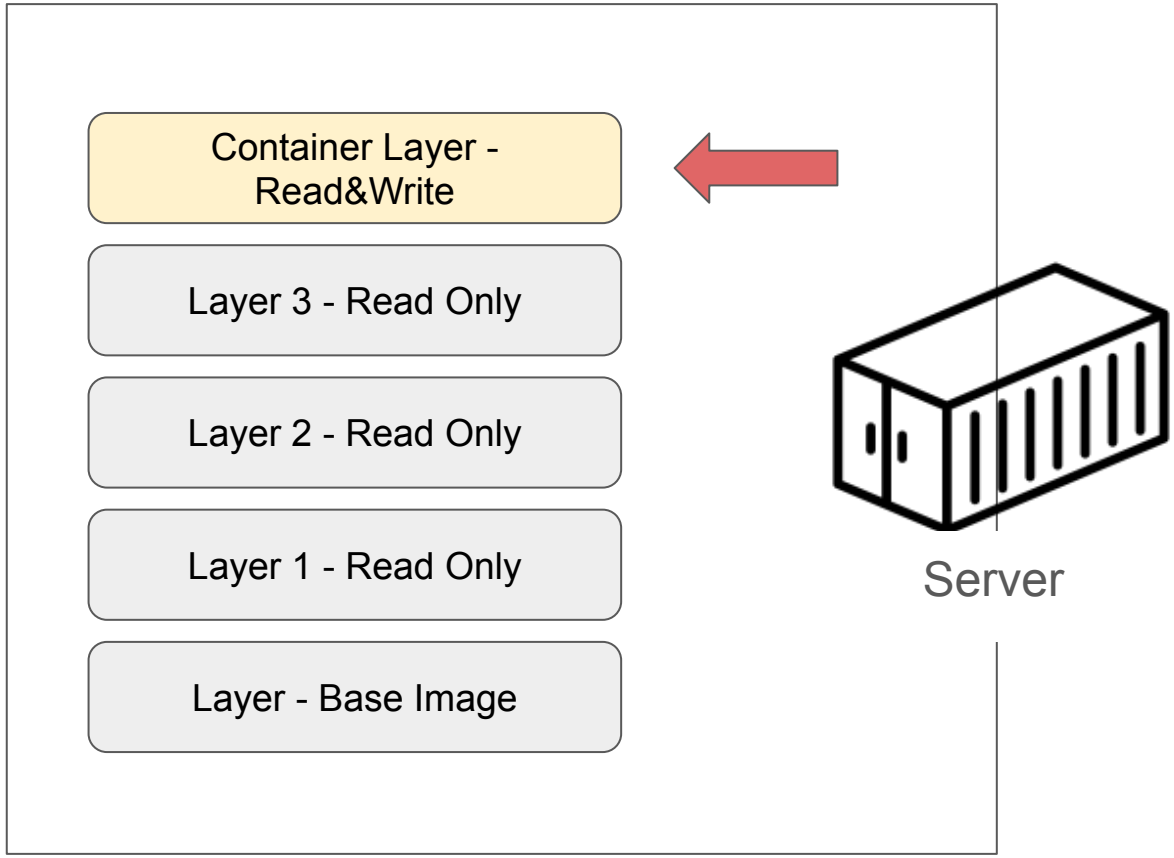

Dockerfile
FROM
Stage 1
COPY
FROM
Stage 2
COPY
FROM
Stage 3
Multi-Stage Dockerfile: Awesome Approach To Optimize Your Container Size 1

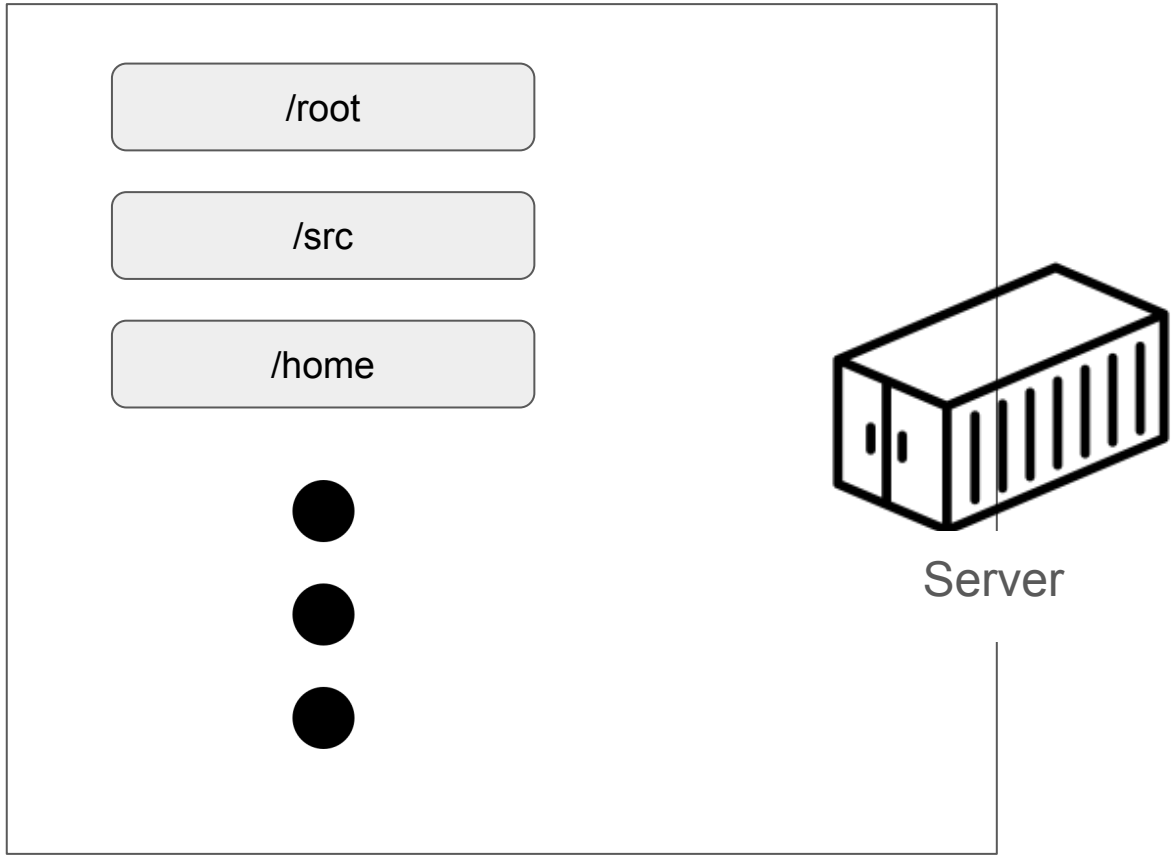
Dockerfile 작성 Tip

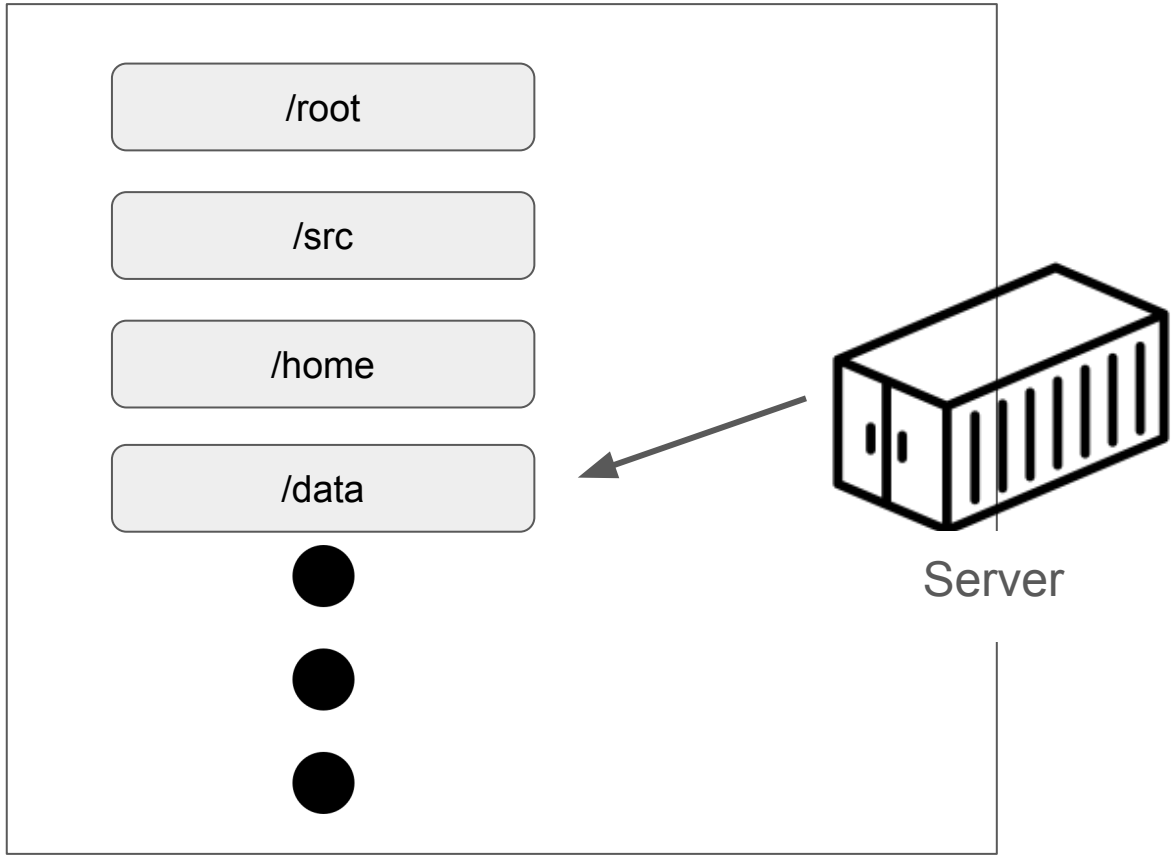
Layers	Cache?
`FROM ubuntu:latest`	✓
`RUN apt-get update \` `  && apt-get install build-essentials`	✓
`COPY main.c Makefile /src/`	✗
`WORKDIR /src`	✗
`RUN make build`	✗

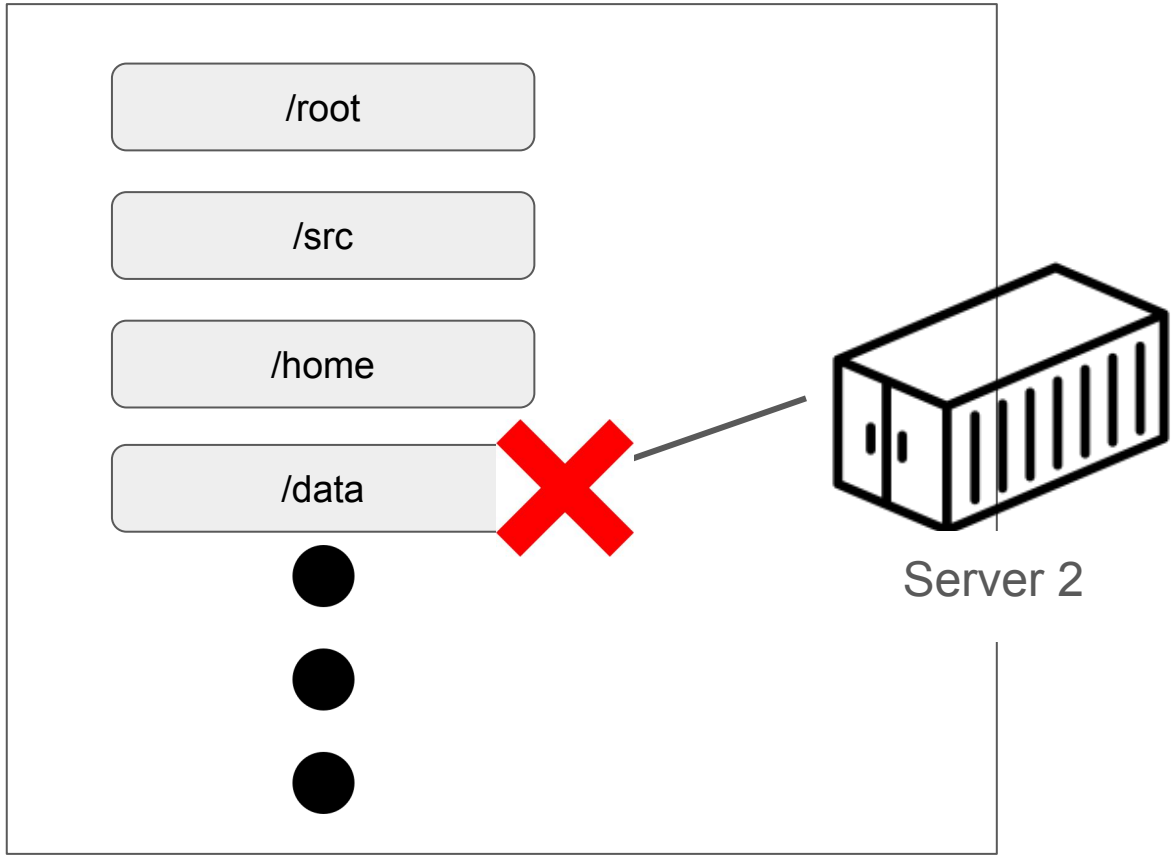
Volume

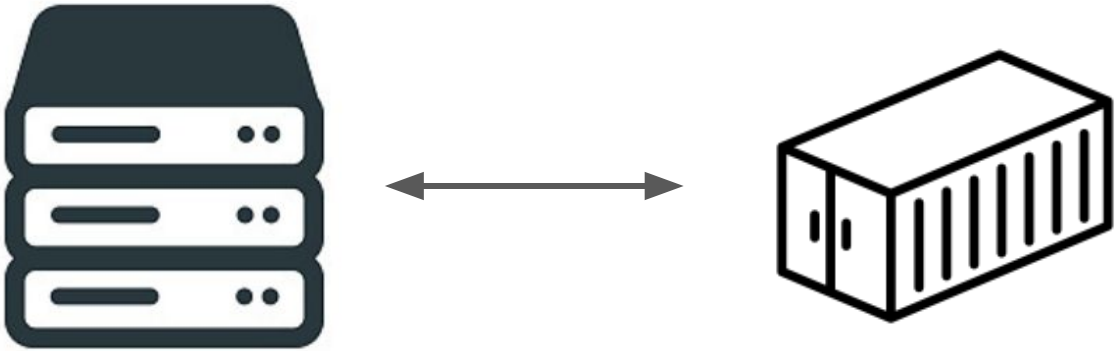

Container Layer - Read&Write
Layer 3 - Read Only
Layer 2 - Read Only
Layer 1 - Read Only
Layer - Base Image
Server


Container Layer - Read&Write
Layer 3 - Read Only
Layer 2 - Read Only
Layer 1 - Read Only
Layer - Base Image
Server


/root
/src
/home
Server

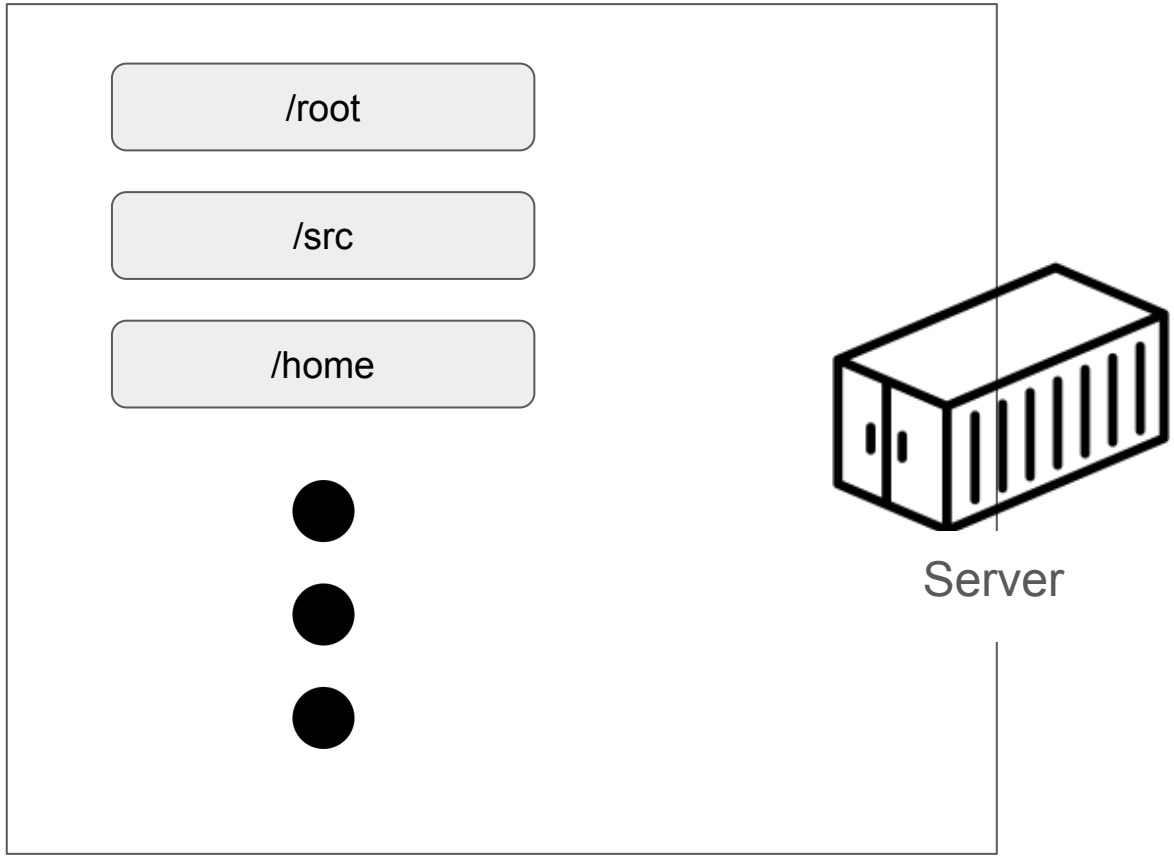

/root
/src
/home
/data
Server

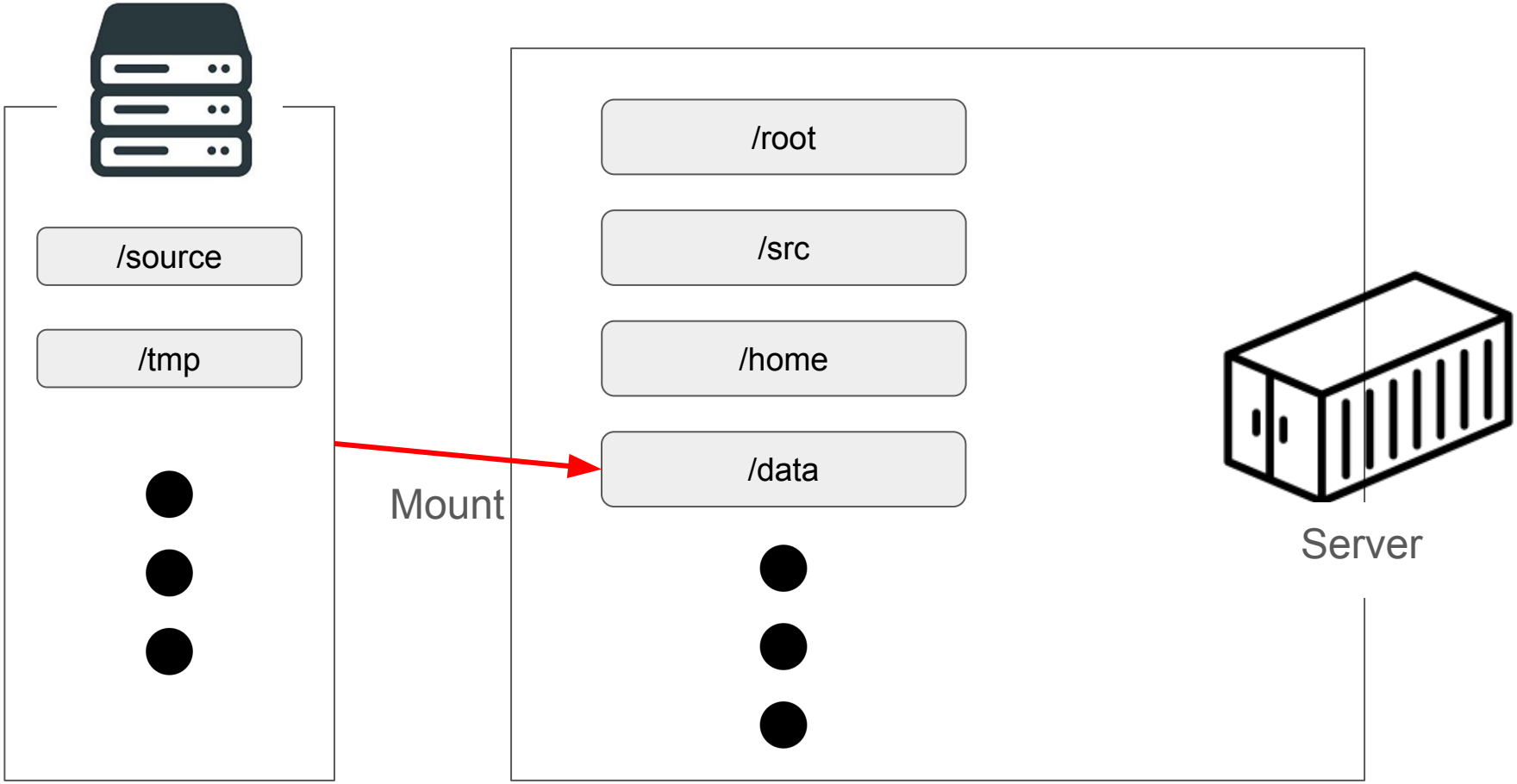

/root
/src
/home
/data
Server 2

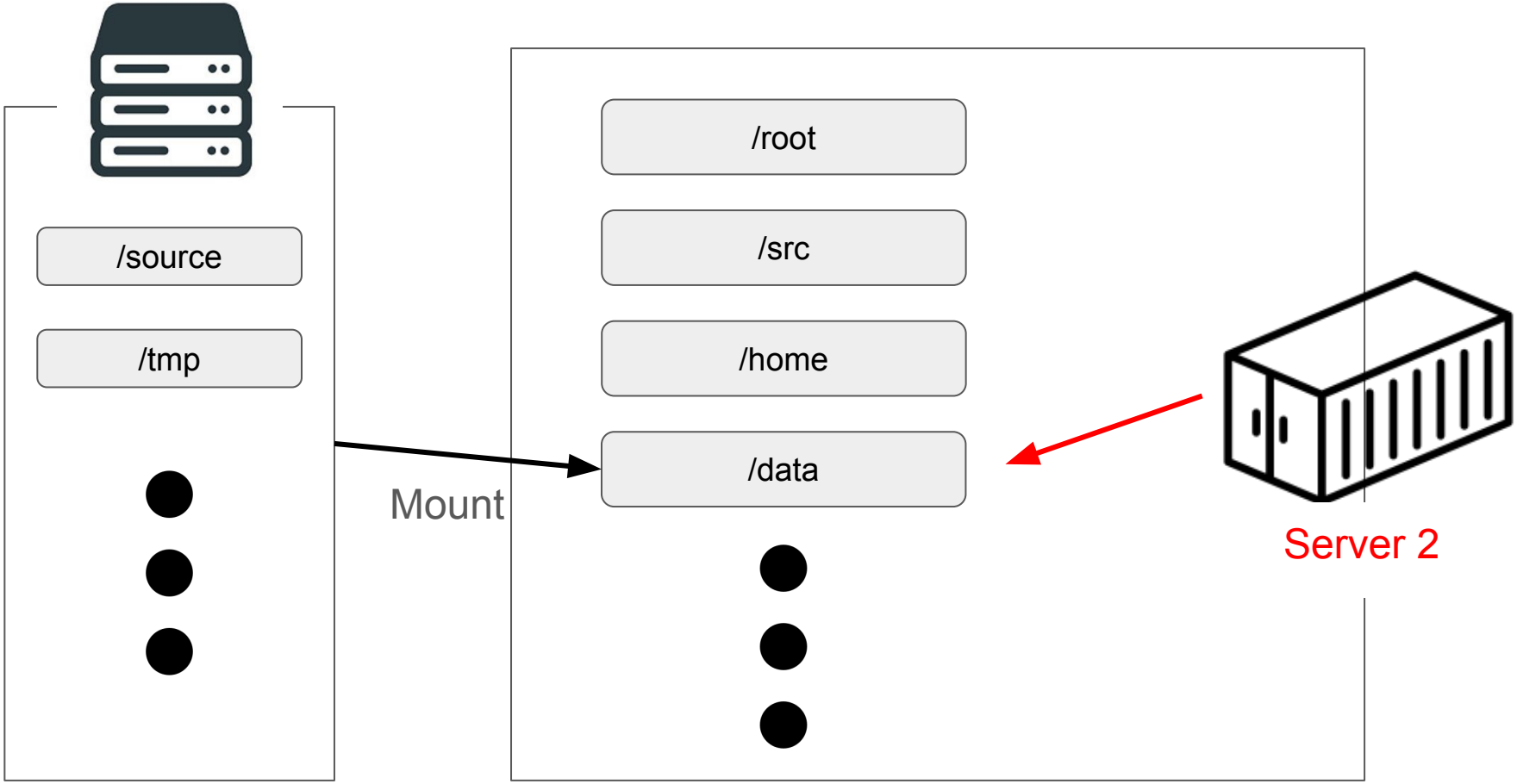


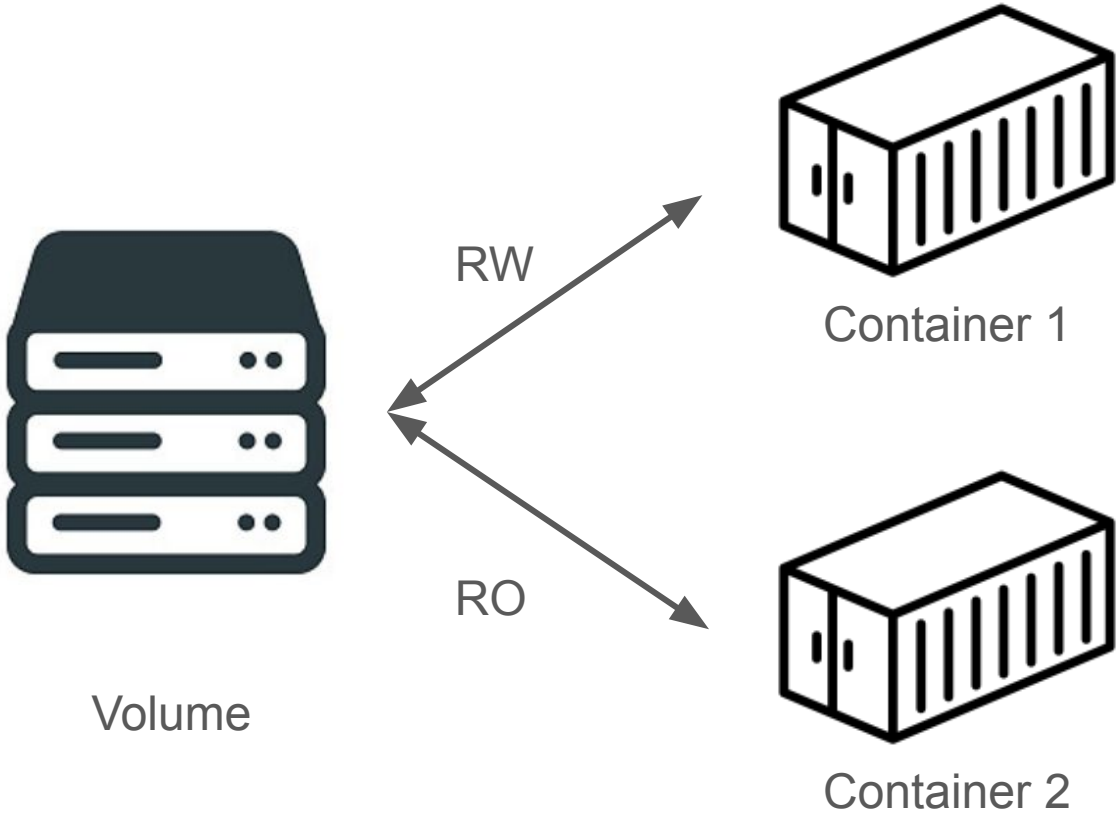
Volume

Container

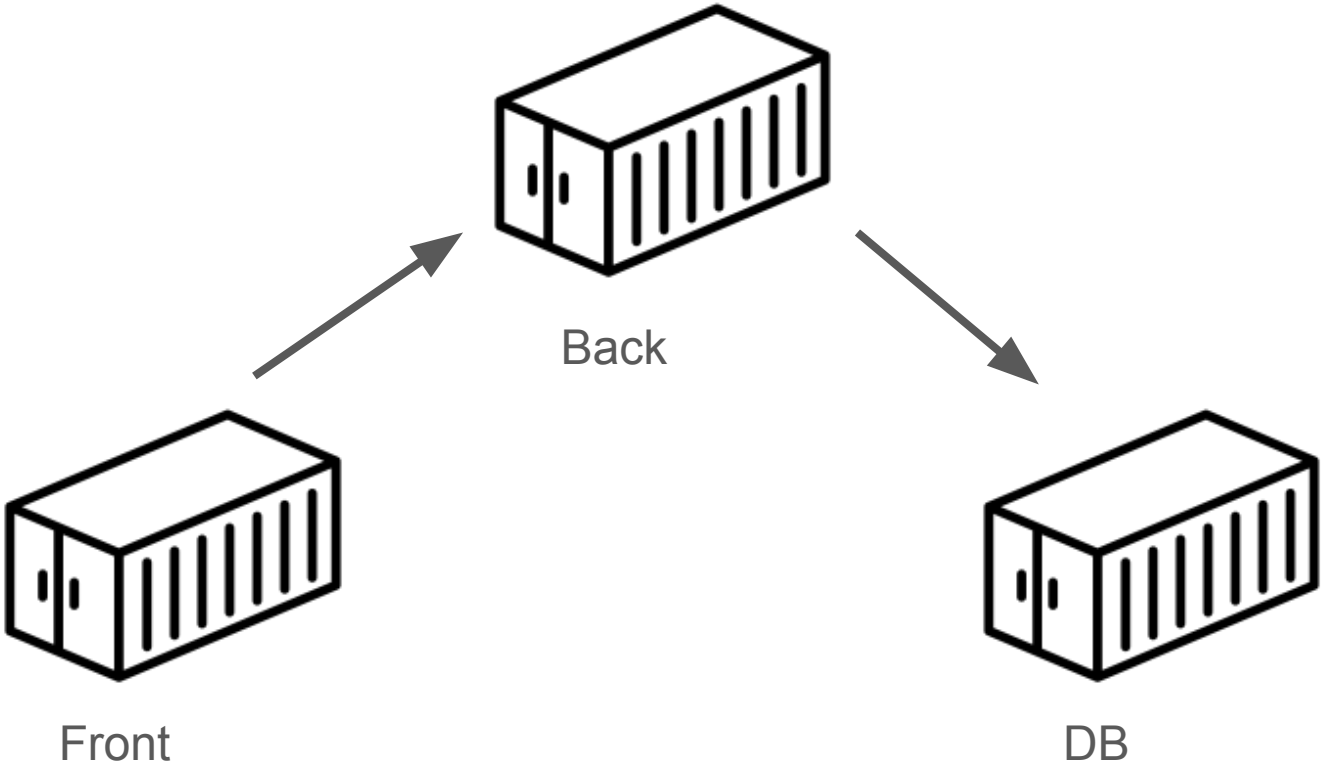

/root
/src
/home
Server

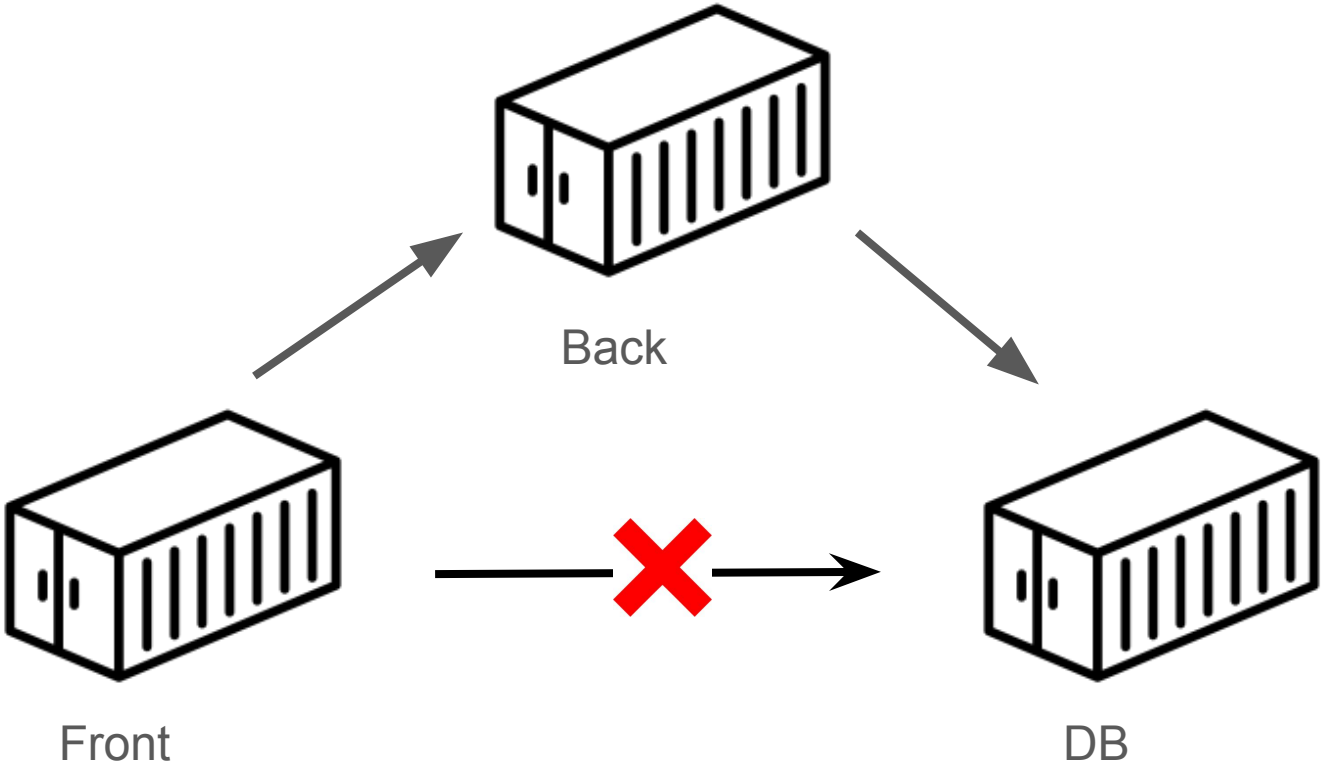

/source
/tmp
Mount
/root
/src
/home
/data
Server

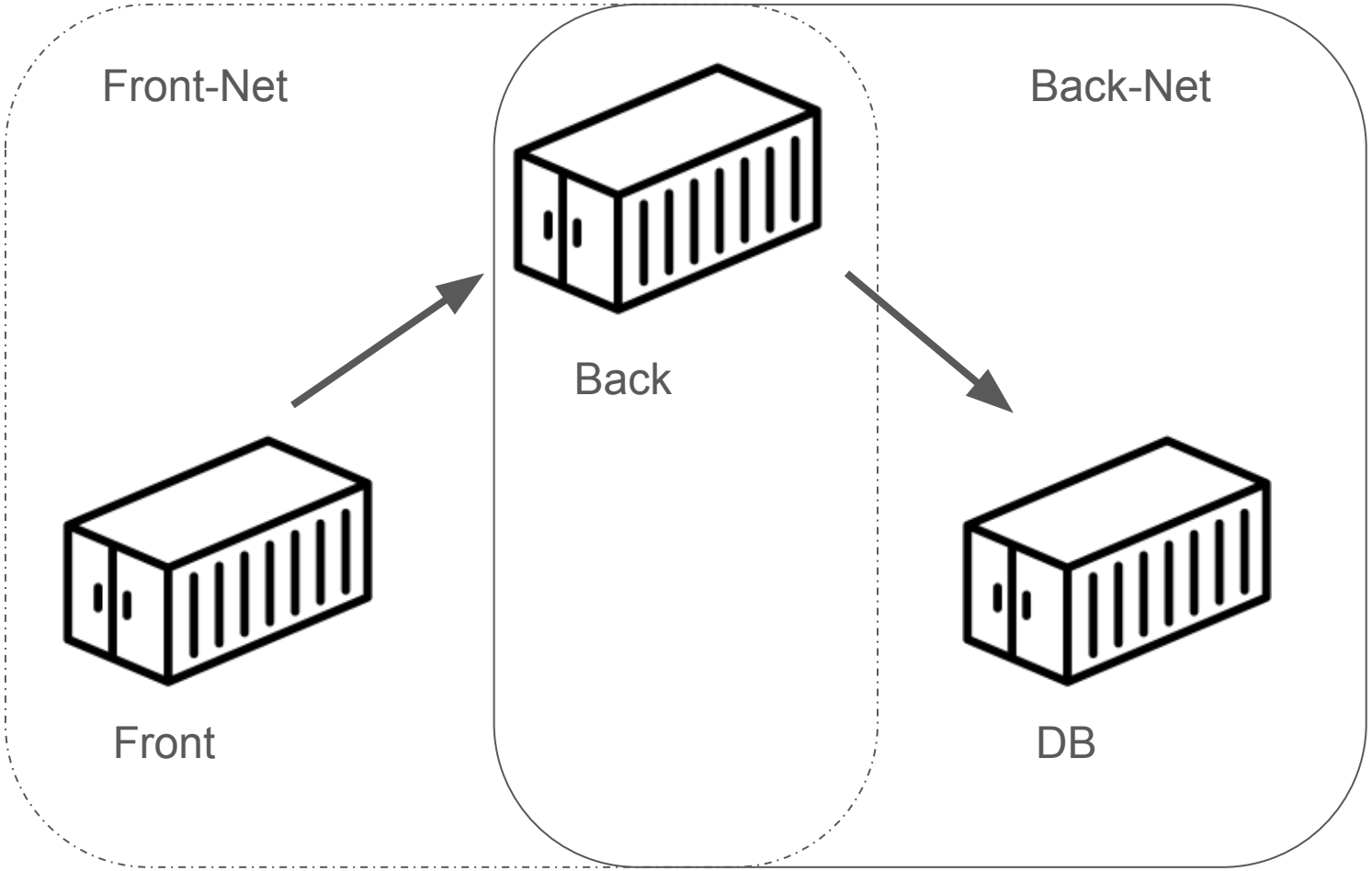

/source
/tmp
Mount
/root
/src
/home
/data
Server 2


RW
Container 1
RO
Volume
Container 2

Network


Back
Front
DB


Back
Front
DB


Front-Net
Back-Net
Back
Front
DB